# The use of software systems to implement Case-Based Reasoning enabled intelligent components for architectural briefing and design

By

Dirk Cornelis Uys Conradie

Submitted in fulfilment of part of the requirements
for the degree of Philosophiae Doctor (Applied Sciences)
Faculty of Engineering, the Built Environment and
Information Technology
University of Pretoria
South Africa

Study Leader/Promotor: Prof. D. Holm

October 2000

# Abstract

This thesis describes the development of a prototype *Case-Based Reasoning* (CBR) enabled intelligent component system, called Architectural General Object System (ARGOS), to facilitate the storage of design information in lightweight cases that can be used on the desktop computer over the total life of the facility. It uses *CBR* techniques combined with *Microsoft ActiveX* controls (object technology) to provide a useful autonomous component to implement some of the software requirements of such a system within the context of the global design and construction environment. These technologies ensure a platform independent environment and integration into the Internet. The use of *XML* (Extensible Mark-up Language) as a design language is explored to facilitate the storage of design data in a persistent and neutral manner independent from the software that originally created it. This ensures a long data life and the enables different actors over the life cycle of a facility to use their own relevant software to process the design information.

During the development of *AEDES* (Architectural Evaluation and Design System), the research team realised that the problem of structuring design knowledge in such a way to support relevant software systems across the life cycle of a facility is far more complex than originally anticipated. Although there are many similarities between the construction and the manufacturing industries, there are also significant and problematic differences. Architectural design tasks take place in an open world where the reasoner's knowledge is incomplete or inconsistent. Due to this the focus in computer-aided architectural design research has shifted back and forth from attempts to totally automate the entire design process to its partial support through drafting tools.

In an attempt to overcome some of the enormous complexities, that researchers struggled with over the past 35 years, a prototype intelligent autonomous design component *ARGOS* is developed in this research. It is clear that automated design methods are not tractable and it is therefore more worthwhile to pursue the creation of a neutral design language and the creation of intelligent and flexible design tools to manipulate these design fragments.

An in-depth study is made of various important out-of-industry manufacturing techniques, *CBR* and object technology and to establish clearly what the desirable characteristics of *ARGOS* should be. An important requirement is that *ARGOS* should be generic and non-prescriptive and should work in a *Microsoft Windows* compliant environment. A solution without the use of *CAD* is proposed that ensure a generic solution that could add value to many different construction industry actors in many different environments. More recently attempts are being made to introduce *post-modern* Artificial Intelligence (AI) into design and architecture. Despite all these efforts it is clear that architectural briefing and design has not reached the status of a science and it is unlikely ever to. This is confirmed by recent breakthroughs in the field of Artificial Intelligence (AI) and Knowledge Management that provide deeper insights into the cognitive processes of the designer.

This study indicates that *XML* is a viable means of expressing design knowledge and a feasible alternative for the complex Building Product Models currently proposed whilst at the same time supporting operations in the Internet environment. Design information and the ability to retrieve it is now more important than the software application that originally created it. The autonomous intelligent component *ARGOS* provides a method to encapsulate design knowledge at both tacit and explicit cognitive levels whilst at the same time providing global communication in a convenient desktop environment. *ARGOS* is designed in a parametric way that supports any design process that requires positional, volumetric and spatial relationship analysis in both 2D and 3D. Multiple autonomous copies can be placed in a container environment such as Excel. Any process written in any computer language that supports the use of ActiveX controls can be used to manipulate the *ARGOS* instances.

# Ekserp

Hierdie verhandeling beskryf die ontwikkeling van ‘n prototipe intelligente komponentstelsel met *Case-Based Reasoning* (CBR) vermoë. Dit word Argitektuur Objek Stelsel (ARGOS) genoem en maak die berging van lewenssiklus ontwerpinligting in kompakte gevalle op ‘n mikrorekenaar moontlik. *CBR*-tegnieke word gekombineer met *Microsoft ActiveX* objektegnologie in die ontwerp van ‘n outonome komponent wat sommige van die programmatuurbehoeftes in die globale ontwerp- en konstruksie-omgewing kan bevredig. Die tegnologieë verseker ‘n platform-onafhanklike uitvoering en gerieflike integrasie in die Internet. *XML* (Extensible Mark-up Language) word as ‘n ontwerptaal gebruik wat die berging van ontwerpinligting op ‘n standhoudende en neutrale wyse moontlik maak, ongeag die programmatuur wat dit oorspronklik geskep het. Dit verseker ‘n lang dataleeftyd en laat verskillende gebruikers oor die lewenssiklus van die fasiliteit relevante programmatuur aanwend om die ontwerpinligting te verwerk.

Gedurende die ontwikkeling van *AEDES* (Architectural Evaluation and Design System), het die span ontdek dat die strukturering van ontwerpinligting, op so ‘n wyse dat dit vir programmatuurstelsels oor die lewenssiklus van ‘n fasiliteit bruikbaar is, aansienlik komplekser is as aanvanklik vermoed. Alhoewel daar heelwat ooreenkomste tussen die konstruksiebedryf en die vervaardigingsindustrie bestaan is daar ook betekenisvolle en problematiese verskille. Argitektuurontwerp vind plaas in ‘n oop wêreld waar die ontwerper (denker) se kennis onvolledig of inkonsekwent is. Derhalwe het die fokus in rekenaar-gesteunde argiteksontwerpnavorsing tussen die uiterstes van totale outomatisasie tot gedeeltelike ondersteuning deur tekenstelsels gewissel.

In ‘n poging om sommige van die enorme kompleksiteite die hoof te bied waarmee talle navorsers oor die afgelope 35 jaar geworstel het, is die outonome ontwerpkomponent *ARGOS* ontwikkel. Dit is duidelik dat geoutomatiseerde ontwerpmetodes nie haalbaar is nie en dat dit dus die moeite werd om eerder die daarstelling van ‘n gerieflike en neutrale ontwerptaal en skep van intelligente en aanpasbare elektroniese ontwerpgereedskap na te strewe wat die betrokke ontwerpinligting kan gebruik.

‘n Omvattende studie word gemaak van verskeie belangrike vervaardigingsindustrie tegnieke, *CBR* en objektegnologie buite die domein van argitektuur om die wenslike karakterestieke van *ARGOS* te bepaal. Een van die belangrikste vereistes is dat *ARGOS* nie-voorskriftelik en in ‘n *Microsoft Windows* aanpasbare omgewing ontplooibaar moet wees. ‘n Generiese oplossing sonder die gebruik van *CAD* word voorgestel sodat dit kan waarde toevoeg tot stelsels wat deur verskillende gebruikers in ‘n wye verskeidenheid van omgewings gebruik word. Tans word verskeie pogings aangewend om *post-moderne Kunsmatige Intelligensie* (KI) in ontwerp en argitektuur toe te pas. Desondanks al hierdie pogings is dit duidelik dat argitektuuropdraggewing en ontwerp nog nie die status van ‘n wetenskap bereik het nie, en waarskynlik nooit sal bereik nie. Dit word bevestig deur deurbrake in KI en kennisbestuur wat diepere insigte in die kognitiewe vermoëns van die kreatiewe ontwerper aan die lig gebring het.

Die studie toon aan dat *XML* ‘n lewensvatbare taal is om ontwerpinligting te struktureer en ‘n alternatief vir die komplekse *Gebou Produk Modelle* (Building Product Models) is wat op die oomblik voorgestel word. Ontwerpinligting en die vermoë om dit te herwin het nou belangriker geword as die programmatuur wat dit oorspronklik geskep het. XML ondersteun ook die Internet. *ARGOS* het metodes om ontwerpinligting van beide stilswyende en eksplisiete kognitiewe aard te verpak. Globale kommunikasie is nou moontlik vanaf die mikrorekenaar. *ARGOS* het ‘n parametriese ontwerp wat ontwerpprosesse in posisie, volume en ruimtelike verwantskaps ontledings in beide 2D en 3D ondersteun. Veelvuldige outonome

kopieë kan in ‘n houeromgewing soos *Microsoft Excel* geplaas word. Enige proses in enige rekenaartaal wat *ActiveX* objekte ondersteun kan *ARGOS* objekte manipuleer.

# Acknowledgements

I would like to acknowledge the valued contributions and support of the following persons and organisations:

- *Prof. Dieter Holm* for his enthusiasm, guidance and support throughout the project;
- *Dr. Ben van Vliet* for his expert guidance and insights on the QFD process;
- *Prof. Craig Zimring, Janet Kolodner*, *Charles Eastman, Ashok Goel*, and *Marin Simina* at the Georgia Institute of Technology for expert guidance and insights into Case-Based Reasoning and the cognitive aspects of design;
- *Kirstin Küsel* for her support, enthusiasm and persistence in the attempts to understand the open world of architectural design better;
- My relatives and friends for their moral support and encouragement;
- My wife Christa for her love and endurance that has greatly contributed to the success of the work presented here.

This thesis is dedicated to my father, who encouraged me to follow a career in the sciences and set an exceptionally high standard for me to follow in dedicated service to his family and South African citizens at large over many years.

# Contents

# List of figures

# List of tables

# Definition of terms

**AEDES**
An acronym for Architectural Evaluation and Design System. This was an early attempt to structure design data during the briefing and design phases to assist with knowledge management across the life cycle of a facility (Conradie *et al*. 1999).

**ARGOS**
An acronym for Architectural General Object System, a Case-Based Reasoning enabled ActiveX intelligent component that can be used in Microsoft compliant container environments such as Microsoft Excel, Word, Access, Visio and Arena.

**Artificial Intelligence (AI)**
In the past definitions such as the following were used:
Luger and Stubblefield defined AI as the branch of computer science that is concerned with the automation of intelligent behaviour (Riesbeck 1996:373).
Minsky defined AI as the field of research concerned with making machines do things that people consider require intelligence (Riesbeck 1996:373).
Charniak and McDermott define AI as the study of mental faculties through the use of computational models (Riesbeck 1996:373).

The definition that is used in the present study is the one of Riesbeck (1996:374) for *Post-Modern AI*. AI is the search for answers to the eternal question why computers are so stupid. In *Post-Modern AI*, the AI becomes an invisible part of the overall system.

**Building Product Model (BPM)**
A BPR is a digital information structure of the objects making up a building, capturing the form, behaviour and relations of the parts and assemblies within the building. A BPR is potentially a richer representation than any set of drawings and can be implemented in multiple ways, including as an ASCII file or as a database (Eastman 1999).

**Blackboard –Based Architecture**
A Case-Based Reasoning architecture that offers flexible, opportunistic control capabilities. A blackboard architecture separates control knowledge from the domain knowledge contained in the knowledge sources (Rissland *et al*. 1991:77-78).

**Case**
A case is a contextualized piece of knowledge representing an experience that teaches a lesson fundamental to achieving the goals of the reasoner (Kolodner *et al*. 1996:36).

**Case-Based Reasoning (CBR)**
CBR solves new problems by adapting solutions that were used to solve old problems. The intuition of CBR is that situations recur with regularity. What was done in one situation is likely to be applicable in a similar situation. If we know what worked in a previous situation similar to the new one, we start with that in reasoning about the new situation (Riesbeck *et al*. 1989:25; Kolodner 1993:8).

**Concept Selection**
Concept selection is the emergence and selection of the best and strongest concepts with respect to customer needs and other criteria. Although creativity is essential throughout the entire product development process, concept selection reduces the number of alternatives under consideration. Concept selection is one of the most critical and difficult problems in design (Pugh 1996:167).

**Constraint**
In order to carry out some design activity, certain information must be available. In addition certain conditions, states or evaluations may apply to the data.

**Critic**
A critic is a piece of software that fires under certain circumstances to alert of possible design conflicts such as a fuel store that is right next to an operating theatre.

**Frame**
A frame is a case-like entity that records relationships between parts of a proposed solution but is more abstract than a case itself. Framing a problem generally means choosing some set of its specifications to concentrate on and deriving a framework that becomes more refined over time (Kolodner 1993:523).

**Fuzzy sets**
Bellman and Zadeh (1970) and Bojadziev *et al.* (1995:113) describe fuzzy sets as a special class of object in which there is no sharp boundary between those objects that belong to the class and those that do not.

**Intelligent Component**
In this view the problem of AI is to describe and build components that reduce the stupidity of the systems in which they function.

**Knowledge Management (KM)**
Knowledge management (KM), as defined by the GartnerGroup, is a discipline with new processes and technologies that differentiate it from information management. New technologies are required to capture knowledge that was previously tacit. Tacit knowledge is embodied in the minds and expertise of individuals. Once captured, knowledge must be shared to leverage its value and reused in similar situations and contexts.

**Object-oriented design**
According to Meyer (1988) Object-oriented design is the method which leads to software architectures based on the objects every system or subsystem manipulates rather than the function it is meant to ensure. Object-oriented design is also the construction of software systems as structured collections of abstract data type implementations (Meyer 1988).

**Open World**
An open world denotes any problem-solving situation in which the reasoner's knowledge is incomplete or inconsistent (Hinrichs 1991:5).

**Quality Function Deployment (QFD)**
QFD is a method for structured product planning and development that enables a development team to specify clearly the customer's wants and needs and then to evaluate each proposed product or service capability systematically in terms of its impact on meeting those needs (Cohen 1995:11).

**Scalable Vector Graphics (SVG)**
A working draft of 29 June 2000 of the W3C defines the features and syntax for Scalable Vector Graphics (SVG), a language for describing two-dimensional vector and mixed vector/ raster graphics in XML. SVG is a language for describing two-dimensional graphics in XML. SVG allows for vector graphic shapes (paths consisting of straight lines and curves), images and text. Graphical objects can be grouped, styled, transformed and composited into previously rendered objects. The feature set includes nested transformations, clipping paths, alpha masks, filter effects and template objects.

**Systems Engineering (SE)**
An interdisciplinary approach and means to enable the realisation of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem.

**Vector Markup Language (VML)**
Microsoft developed their own XML application for vector graphics called VML. VML is more finished than SVG and is already supported by Internet Explorer 5.0 and Microsoft Office 2000. VML is not as ambitious as SVG and leaves out advanced features such as clipping and masking.

**XML**
XML was developed by an XML Working Group (originally known as the SGML Editorial Review Board) formed under the auspices of the World Wide Web Consortium (W3C) in 1996. XML is a set of rules for defining semantic tags that break a document into parts and identify the different parts of the document. It is also a meta-markup language that defines a syntax used to define other domain-specific, semantic structured mark-up languages (Harold 1999:3).

# List of abbreviations

| | |
|---|---|
| ADE | Application Development Environments |
| AEDES | Architectural Evaluation and Design System |
| AI | Artificial Intelligence |
| ARGOS | Architectural General Object System |
| BEARS | Building Environmental Assessment and Rating System for South Africa |
| BMMS | Building Maintenance Management System |
| BOMSIG | Business Object Model Special Interest Group |
| BPM | Building Product Model |
| CASE | Computer-Aided Software Engineering |
| CBD | Case-Based Design |
| CBR | Case-Based Reasoning |
| CBT | Computer-Based Training  and Teaching |
| CE | Concurrent engineering (CE) |
| CKO | Chief Knowledge Officer |
| COM | Component Object Model |
| CONSENS | Concurrent Simultaneous Engineering System |
| CORBA | Common Object Request Broker Architecture |
| CPDM | Common Product Data Model |
| CSS | Cascading Style Sheets |
| DBMS | Database Management Systems |
| DCE | Distributed Computing Environment Group |
| DCOM | Distributed Component Object Model |
| DTD | Document Type Definition |
| DXF | Data Interchange Format |
| EQFD | Enhanced QFD |
| FFE | Fuzzy Front End |
| FM | Facilities Management |
| FMEA | Failure Mode and Effects Analysis |
| GUID | Global Unique Identifier |
| HOQ | House of Quality |
| HTML | Hypertext Mark-up Language |
| IAI | International Alliance for Interoperability |
| IDC | International Data Corporation |
| IGES | Initial Graphics Exchange Specification |
| IR | Information Retrieval |
| IS | Information Science |
| ISO-STEP | International Standards Organisation – Standard for the Exchange of Product model data |
| KA | Knowledge Architect |
| KBCAAD | Knowledge Based Computer-aided Architectural Design |
| KBDS | Knowledge-based Design Systems |
| KBS | Knowledge Based System |
| KE | Kansei Engineering |
| KE | Knowledge Engineering |
| KES | Kansei Engineering System |
| KM | Knowledge Management |
| KMS | Knowledge Management System |
| LTM | Long Term Memory |
| MBR | Model-based Reasoning |
| MIT | Massachusetts Institute of Technology |
| MOP | Memory Organisation Packet |
| NGM | Next Generation Manufacturing Company |

| | |
|---|---|
| NLP | Natural Language Processing |
| ODB | Object-Oriented Database or Object Database |
| ODBC | Open Database Connectivity |
| ODL | Object Description Language |
| OE | Operational Expense |
| OID | Object Identifier |
| OLAP | On-line Analytical Processing |
| OLE | Object Linking and Embedding |
| OMG | Object Management Group |
| OOCAD | Object-Oriented Computer Aided Design |
| OOL | Object-Oriented programming languages |
| ORB | Object Request Brokers |
| PDES | Product Data Exchange using STEP |
| PDM | Product Data Modelling |
| PREMIS | Professional Real Estate Management Information System |
| PROCAP | Procedural Guide for Clients, Architects and Other Professionals |
| QA | Quality Assurance |
| QC | Quality Control |
| QFD | Quality Function Deployment |
| RBR | Rule-Based Reasoning |
| ROI | Return on investment |
| SCM | Service Control Manager |
| SD | Semantic Differential |
| SE | Systems Engineering |
| SGML | Standard Generalised Mark-up Language |
| SME | Subject Matter Experts |
| SQC | Statistical Quality Control |
| SQL | Structured Query Language |
| SVG | Scalable Vector Graphics |
| TOC | Theory of Constraints |
| TOP | Thematic Organisational Packet |
| TQM | Total Quality Management |
| TQM | Total Quality Movement |
| UDE | Undesirable Effects |
| UIF | Universal Index Frame |
| UR | User Requirement |
| VBA | Visual Basic for Applications |
| VE | Value Engineering |
| VML | Vector Mark-up Language |
| VOC | Voice of Customer |
| VR | Virtual Reality |
| VRML | Virtual Reality Mark-up Language |
| W3C | World Wide Web Consortium |
| WM | Working Memory |
| www | world wide web |
| XML | Extensible Mark-up Language |
| XSL | Extensible Style language |

# Chapter 1: Introduction and overview

This chapter provides a brief introduction to the impact of the knowledge-age drivers on the product development processes. This suggests that the construction industry need to be improved in general and architectural briefing and design specifically if it is to remain competitive. It identifies the fact that, although there are similarities between product development and architectural design, there are also significant and problematic differences. Realistic Artificial Intelligence in a *post-modern* form brought new opportunities to improve architectural design activities.

The advent of the information age and a knowledge economy brought a necessity for global competitiveness and a need for product innovation (TechnoSolve 1998:1).

The Global drivers of the new marketplace are (Agility Forum 1997):

- Ubiquitous availability and distribution of information.
- Accelerating pace of change in technology.
- Rapidly expanding technology access.
- Globalisation of markets and business competition.
- Global wage and job skills shift.
- Environmental responsibility and resource limitations.
- Increasing customer expectations.

The Next Generation Manufacturing Company (NGM) will succeed through the integrated performance of people, business processes and technology. Of the NGM imperatives identified *Rapid Product/ Process Realisation* is the aspect that is most relevant to the present study.

This challenges firms to:

- Address smaller market segments, with an increased variety of products.
- Increase the frequency of product introductions.
- Compress the lead-time for product development.
- Shorten product life cycles.
- Increase product complexity.
- Distribute product development and production activities across a network of firms instead of within a single integrated firm. (TechnoSolve 1998:1).

These challenges imply the need to design and develop families of products by networks of firms on compressed development schedules, which in turn implies increasingly more knowledge from the designer regarding the life cycle costs of a product at the conception of the design problem, where few decisions have been made and little cost committed to production. Designers are under increasing pressure to apply upstream design techniques to improve quality, whilst decreasing downstream costs. *“ Problems experienced downstream are symptoms of neglect upstream. Upstream problems can only be solved upstream. The ability to influence a system’s characteristics diminishes very rapidly as the system proceeds from one phase of its life cycle to the next.”* (Sparrius 1998:1-2) (Figure 1).

In order to implement these challenges, companies, especially in the manufacturing industry, are re-engineering their business processes to survive in the new knowledge economy. Traditional manufacturing development processes are more and more re-engineered towards a concurrent engineering model, tailor made for the requirements of the information age. To accommodate business shifts and an ever-changing social environment, it is important to emphasise the importance of a holistic perspective in these re-engineering exercises. Re-engineering should include setting up an integrated organisation information environment, an organisational Information Ecology (Davenport 1997:4).

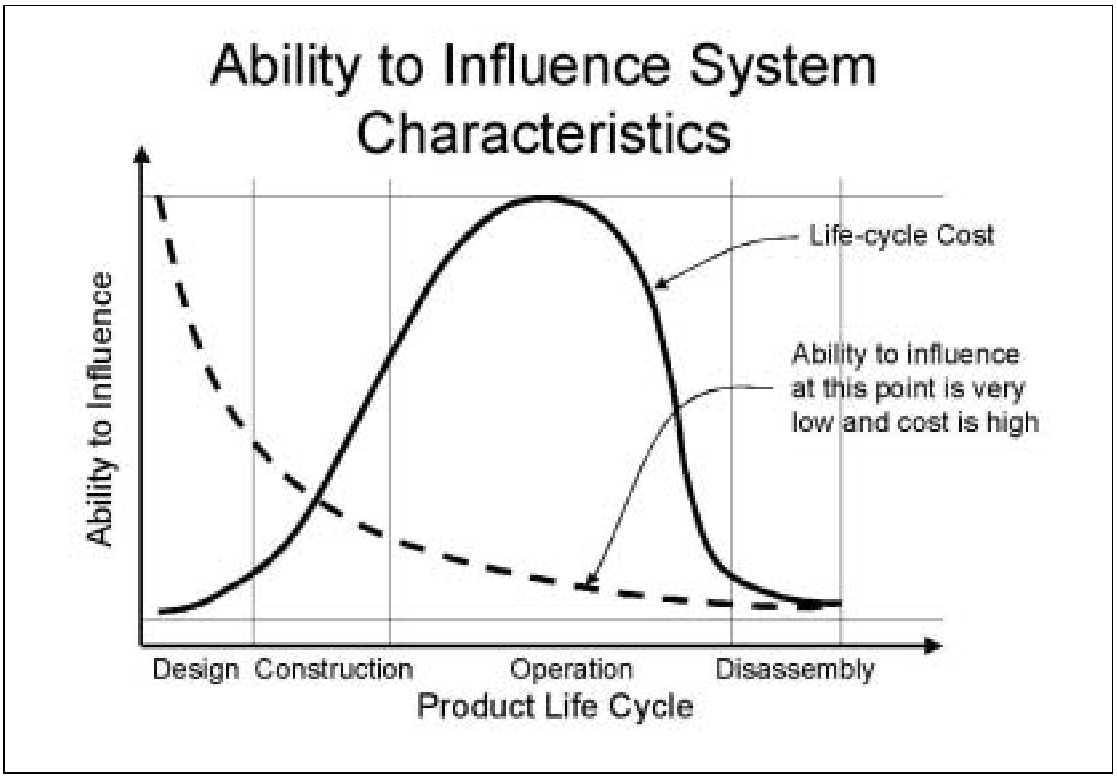


Figure 1: Ability to influence system characteristics (Sparrius 1998: 1.1)

Simultaneously to the abovementioned, the advent of *Post-Modern Artificial Intelligence* (AI) brought new opportunities in the design field. According to Riesbeck (1996:374) AI is the search for answers to the eternal question: Why are computers so stupid? Riesbeck (1996:377) indicates that the problem of AI is to describe and build components that reduce the stupidity of the systems in which they function. The goal should be the improvement of how systems function through the development of intelligent components to those systems. For example, one does not want an automated designer. One wants a design support facility that is not stupid. One requires one that knows concepts, not keywords, that will be able to retrieve relevant design information when the designer is designing a new departure lounge of an airport. In *Post-Modern AI* , AI becomes an invisible part of the overall system.

Although architectural practitioners have not traditionally been seen as being in the business of manufacturing products, research publications (Pugh 1996), (Anumba *et al.* 1997:8) indicate strong similarities between the building construction industry and the manufacturing industry. Not only are both industries in the business of designing and building physical objects (products or artefacts) to satisfy specific client (customer) requirements in a specific environment, but they are also following similar design-build processes. However there are also significant and problematic differences.

Architectural design tasks take place in an open world (Hinrichs 1991:5). An open world denotes any problem-solving situation in which the reasoner's knowledge is incomplete or inconsistent. A design problem solver's knowledge can be incomplete in several ways (Hinrichs 1991:5):

- *Open categories*. An open category is one for which membership cannot be determined from lack of knowledge of membership. The set of primary colours is a closed set.

However the set of possible solutions for a particular architectural design problem is open.

- *Incomplete domain theories*. In architecture the designer's theory of his domain is incomplete. He may not know or be able to retrieve all the causal relationships and facts relevant to a given problem. Reasoning processes that demand theorem proving are not possible.
- *Under-specified problems*: Design problems that are under-specified have solution criteria that are incomplete. This means that the class of solutions forms an open category. If a problem is under-specified the solution needs to be satisfactory rather than absolutely correct. To determine when an architectural design solution is adequate is critical.

According to Hinrichs (1991:15-16) the generation of plausible solutions for a design problem is not simply tedious, it is downright problematic. The generation of a solution is difficult for the following reasons:

- *The design problem spaces are not enumerable*. As with irrational numbers, there is no function that permits a designer to generate the next possible design. Because the problems are ill structured in this way, the designer must supply his own structure in terms of a vocabulary of categories of designs and design components. The content and organisation of this design knowledge is critical for the task of generation. This indicates the necessity for a convenient, flexible and structured language to express designs in.

- *Design constraints are not constructive*. The constraints of a design problem do not directly suggest solutions. Constraints on most design problems rule out an infinite number of possible designs and possibly still permit infinitely many. Due to this the designer needs some kind of associative mechanism to identify members of the categories.

- *Categories in the design vocabulary are fuzzy and subjective*. Design problems are not described by necessary and sufficient conditions, but by experience and expectations. The design solutions of today is contextually, functionally and normatively characterised by designs we have known from the past. This strongly indicates that creative designs should be generated from experience and not deductive rules. This seems to question one of the basic antihistoric tenets of modernism.

- *Problems may be barely decomposable*. While it is sometimes possible to break problems into smaller subproblems, these subproblems tend to be highly inter-constrained. Design time and search can be reduced by using entire plausible solutions that are already internally consistent. These designs can be modified when required. If it is not possible to solve the design problem at this higher level of granularity then the problem can be decomposed. Consequently, the design problem solver should be able to retrieve possible solutions in large design knowledge fragments or cases.

- *Design occurs in multiple problem-spaces*. Because it is sometimes necessary to decompose a problem the designer must work at different granularities or problem spaces, such as the design of a door or an entire airport. All indications are that electronic design knowledge should be hierarchically organised rather than relationally.

Over the last twelve years the author built up considerable experience in the design, programming and implementation of Facilities Management (FM) systems that integrates the long-term strategic and short-term operational maintenance. Most of the present FM systems essentially only handle the operational part of the building life cycle. The integration of processes and information between all the important stages of the life cycle of structures is presently not well understood. The National Health Facilities Audits undertaken since 1996 brought the opportunity to implement quantification of observed phenomena. The condition and suitability of every government hospital in the RSA were analysed and quantified in the form of coloured grid, bar graphs and block plans (Conradie 1996). Recent movements and requirements both locally and abroad indicate that FM will increasingly begin to cover the entire life cycle of the building to ensure informed holistic and appropriate decisions. There is also an increased realisation that decisions taken during the briefing and design phase significantly influences the subsequent phases. Unfortunately these decisions sometimes have to be taken with very little software support or the lack of structured design information.

Attempts were made to represent the properties of objects and to modelling and visualise their forms. Design solutions were synthesised and their specific performances evaluated (Carrara *et al.* 1994).

1. Key performance requirements of an Information Age development process (an information management process, integrated with a structured development process) are:

- An integrated life cycle process.
- A concurrent multimedia environment.
- Life cycle requirement validation.
- Storing of structured design knowledge.
- Ad-hoc queries and reports.

2. The architecture for the proposed Information Age process should be designed as part of an organisational Information Ecology, one that *"emphasises an organisation's entire information environment. It addresses all of a firm's values and beliefs about information (culture); how people actually use information and what they do with it (behaviour and work processes); the pitfalls that can interfere with information sharing (policies); and what information systems are already in place (yes, finally technology)."* (Davenport 1997:4).

The above mentioned already suggests that the improvement of architectural briefing and design is not a trivial matter. The successful solution will be an appropriate synthesis of many different techniques. The subsequent chapters will study well-established techniques from the world of manufacturing, the characteristics of design and Artificial Intelligence (AI) in an attempt to understand the characteristics of the early phases of architectural design better and to discover if a significant improvement can possibly be made.

# Chapter 2: Motivation

## 2.1 Problem statement

The purpose of this study is *firstly* to study well-established techniques from the world of manufacturing, the characteristics of design and Artificial Intelligence (AI) to understand the characteristics of the early phases of architectural design better and to discover if a significant improvement can possibly be made. *Secondly* an attempt will be made to establish a simple design language to support the life cycle of a construction and *thirdly* to build a prototype design processor that could use the design information.

## 2.2 Sub-problems

**Sub-problem 1**: Can design requirements be sufficiently structured in functions that lead to design elements and specifications to facilitate the storing of design knowledge?

**Hypothesis**: A building can be seen as a production product and hence established Systems Engineering techniques and quality control measures can be applied to the briefing and design process.

**Assumption:** Due to the high cognitive content of design it is assumed that techniques from the manufacturing industry will only partially solve the problem and therefore a bridging technique (non-prescriptive) between the capabilities of the human brain and systematic approaches will have to be established.

**Sub-problem 2:** Can a flexible, multimedia database structure that addresses the total life cycle requirements of a building be created using existing software?

**Hypothesis:** The architectural briefing and design process can be structured in such a way that it can be implemented on a software system to ensure total life cycle design.

**Assumption:** The structuring and storage of design information will become more important than the software application that originally created it. Although very complex Building Product Models exist at this stage an attempt will be made to use the technologies offered by the Internet.

**Sub-problem 3:** Can software object technology be used to store architectural designs in such a way as to expedite future designs.

**Hypothesis:** Architectural designs and design parameters can be quantified and electronically packaged in such a way as to expedite future designs that require similar designs or parts of designs. Concurrent briefing and design processes can be implemented on the www within a multi-disciplinary team on a global basis.

**Assumption:** Integration and structuring of the structured multi-media design information is essential if global competitiveness is to be achieved. It is further assumed that this should be the basis on which organisational processes should be built.

## 2.3 Bounds and constraints

It is assumed that the present theoretical basis in diverse fields such as software object technology, Systems Engineering, QFD, TRIZ, Kansei Engineering and CBR is sufficiently developed to enable the implementation on desktop based software system operating in a client-server mode or an Internet based Knowledge Portal. Such a system should be implementable on present desktop computers using Microsoft Windows 95, 98 or NT and standard hardware. The project will not attempt to develop a full commercial system, but will concentrate on a framework and certain sub-modules to illustrate the principles due to financial and time constraints.

The following strategic assumptions are made:

- Microsoft products such as the Windows operating systems, office integration products, object technologies and Internet browsers will remain influential in the short to medium term.
- The Internet/ World Wide Web will be the main information network in the world and the preferred infrastructure for global e-commerce and data exchange.
- Hypertext Mark-up Language (HTML) and Virtual Reality Mark-up Language (VRML) will continue to dominate the www. The new Extensible Mark-up Language (XML) standard as defined by the World Wide Web Consortium 10 February 1998 (http://www.w3.org/TR/1998/REC-xml-19980210) provides a useful basis for the implementation of a flexible and neutral design language. The co-existence of diverse and distributed sources of design knowledge at different levels of specificity rather than a centralised object store.
- Internet based subscriber services will become prominent. This is confirmed by Internet service providers such as ZoomON (http://www.zoomon.com) and Autonomy (http://www.autonomy.com).
- Java, Visual Basic Script (VB Script) and Visual Basic will be the language of choice for Internet applications (Bouzeghoub *et al.* 1997; Lomax 1997).
- Microsoft ActiveX will gain more prominence than CORBA (Lomax 1997).

## 2.4 Research method

This research attempts to create a prototype generic software tool that could aid the early difficult and conceptual stages of design whilst at the same time aiming as low as possible. Aiming low implies the creation of a non-prescriptive affordable tool that can readily fit into any Microsoft Windows compliant container environment, integrated into third party software or be used directly in the Internet. The tool should be usable on the desktop and should be non-CAD centric. The prototype software system will be implemented by means of existing software techniques and products such as Microsoft Internet Explorer, Visual Basic and Microsoft Personal Web Server. Object oriented technology will be used as far as possible.

Simultaneously well-established techniques such as Business- and Systems Engineering, Kansei Engineering, Fuzzy Sets, QFD, Taguchi Techniques, TRIZ and Case-Based Reasoning will be used as a reference framework for the prototype software system.

The results of a recent extensive QFD exercise from a cross-section of construction professionals in the South African construction industry will be used as a means to guide the general direction and characteristics of the generic software tool mentioned above. (Küsel 2000).

Over the past 35 years commercial CAD systems have had little impact on the early, conceptual stages of design. This is the phase where the maximum benefit over the life cycle can be realised at the minimal cost. This inadequacy is further exacerbated by the pressing need to follow a total life cycle approach to architectural briefing and design. Eastman (1994:95, 1999) indicates how long efforts to develop integrated backend databases to support architectural design and construction have been going on. Except in special cases, these efforts have not been very successful.

New product design and development paradigms have emerged in other fields of expertise, yet no total design system exists to address the high level of design complexity in a global architectural environment, one that uses the world wide web (www) without compromising aesthetics and ethics. There is also a clear indication that Knowledge Management (KM) is becoming very prominent.

# Chapter 3: Review of literature

## Introduction

In this chapter the various well-established techniques from the world of manufacturing, knowledge management, knowledge based design, Case-Based Reasoning, objects, Kansei Engineering, Quality Function Deployment and TRIZ are analysed in depth in an attempt to discover if improvements can be made to the early phases of design.

It is not intended to criticise the product innovation methodologies discussed below and mapped out in Figure 2. The intention is rather to establish what value these methodologies could add to the initial and also subsequent stages of the design process. At this stage it is also presumed that the eventual solution proposed in this study should be such that any external product innovation methodology (application) can be applied to it whenever desired.

It is beyond the purposes of this study to provide a detailed discussion of the numerous different design theories.

Over the years many different product development techniques evolved in the world. None of these product innovation methodologies or techniques is capable of solving all the problems inherent in the product design process. However the rich set of techniques is successful in solving many of the sub-aspects.

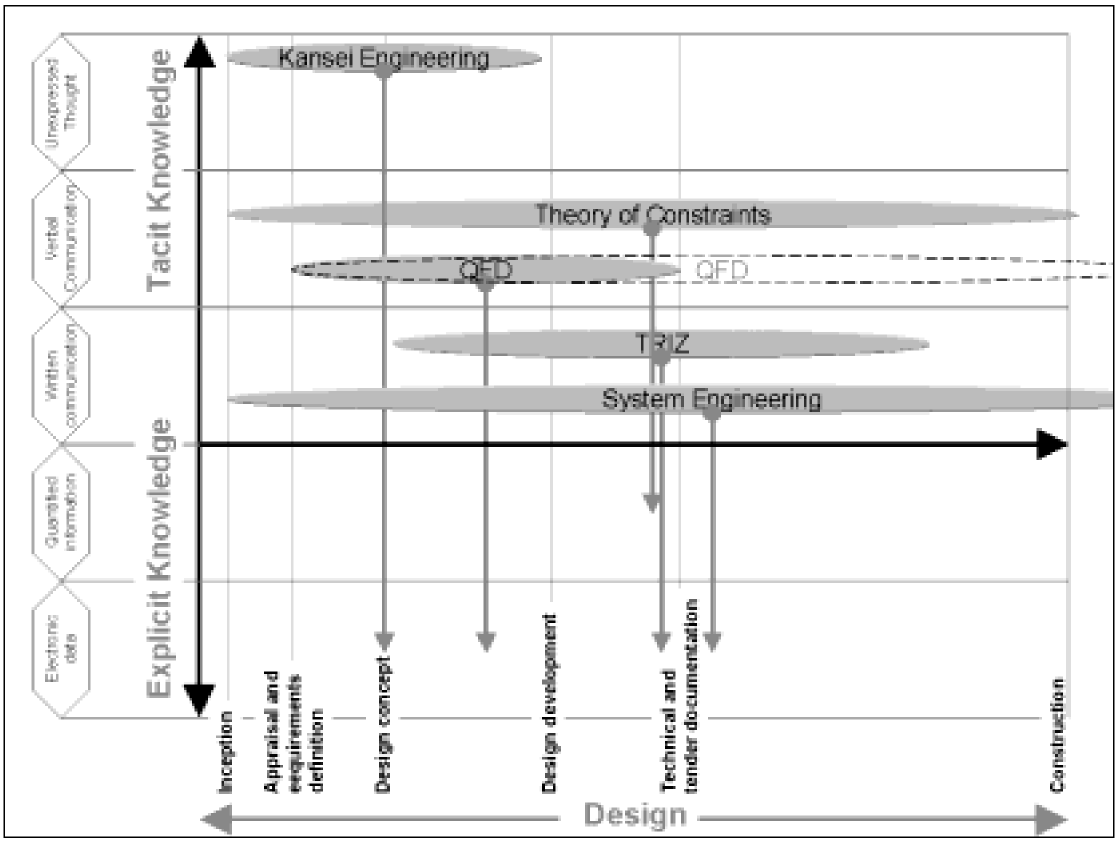


Figure 1: Product innovation methodologies (Collated by author)

The author is of the opinion that the great difficulty that is experienced is partly due to the fact that technology, especially electronic computing, exploded beyond all recognition. This has the effect that most problem solvers are almost by default attempting to solve the particular problems by means of computer models, rules or cases. This creates the incredibly difficult problem that design knowledge must first be quantified or moved from the tacit level to the explicit level (Figure 1) to be in a readily processable form on the electronic computer.

The following current techniques exist that could assist the product development team in product innovation (Figure 1). Although many more techniques exist the present study concentrates on *Case-Based Reasoning* against the background of *Theory of Constraints*, *QFD*, *TRIZ* and *System Engineering*. The assumption with sub-problem 1 was that techniques from the manufacturing industry would only partially solve the problem and that a bridging technique between the human brain and systematic (structured) approaches will have to be established. CBR is specifically studied in the light of sub-problem 3 to see whether designs can use experience from the past to expedite present designs.

Figure 1 maps the different tacit levels where these techniques source information and indicate how far the information can be quantified or moved to a level of explicit knowledge. On the horizontal axis the various main stages of the architectural design process are mapped out. Traditionally QFD was only really useful in the initial design stages up to design development. However in the prototype system *AEDES* attempts were made to stretch it further across the life cycle of the structure, hence the dotted line. *Natural Language Processing* (NLP) is an evolving technology that could greatly assist to turn the spoken and written word into explicit knowledge. Attempts are currently been made to develop NLP software to facilitate concept extraction out of text. However it is going to take at least another three years before a sufficient level of reliability is reached.

The accurate extraction of customer needs are very important for ultimate success of the architectural design. Some methods for defining customer needs are (Zultner 1999):

- *Kansei Engineering* – emotions of the customer.
- *QFD* – voice of the customer.
- *Theory of Constraints* – mind of the customer.
- *Customer Context Analysis* – context of the customer.
- *Systems Dynamics* – environment of the customer.

Methods for structuring product acquisition

- *Business Engineering* – attempt to structure entire product environment.
- *Systems Engineering* – structured methodology for sequencing design activities.

In order to create a digital representation of an architectural design artefact it is necessary to create a building product model. A building product model is a digital information structure of the objects making up a building. It captures the form, behaviour and relations of the parts and assemblies within the building. Major efforts are being made throughout the world to develop such a representation. Among the industry groups involved in achieving this are the U.S.A. based Product Data Exchange using STEP (PDES) organisation and its international counterpart, International Standards organisation – Standard for the Exchange of Product model data (ISO-STEP). Efforts are also being undertaken by research groups funded by the European Union in Europe and by the National Science Foundation in the U.S.A. Another significant effort is of the International Alliance for Interoperability (IAI). According to Eastman (1999) building product models will eventually be used by most people associated with the building and real estate businesses such as architects, engineers, contractors, owners and facility managers.

In this study the use of the structured hierarchical ASCII standard, XML will be used to explore the creation of a simple, yet powerful design language to support desk top based design tools such as design modellers whilst at the same time maintaining a close relationship with the Internet. The opinion is expressed that the ASCII nature of XML makes it a strong candidate for an application independent design language because it is simple but at the same time flexible and extendable. It is also a very powerful integrator of non-XML data. XML provides three constructs that could be used to achieve flexibility and extendibility:

- Notations
- Unparsed external entities
- Processing instructions.

## 3.1 Knowledge management

### 3.1.1 Introduction

This section is included because architectural design depends to a large extent on the availability of sound knowledge. Some of the wide range of knowledge that is required in this domain can be summarised as:

- National Building Regulations.
- Characteristics of materials.
- Construction components.
- Construction methods.
- Energy use.
- Surveyor General's diagram.
- Acoustics.
- Solar movement.
- Anthropometrics.
- Climatic conditions such as temperature, wind and rainfall.
- Construction detail.

If this type of information cannot be readily obtained then it makes the designer ineffective. It is highly desirable that the designer is able to quickly retrieve this knowledge whenever required. Whatever solution is ultimately proposed its success will depend to a large extent on the convenient access to this type of information. With the advent of large construction projects it is also crucial that designers are able to collaborate globally. Microsoft's approach to KM is studied to establish if it is suitable for the proposed solution.

Nonaka (1998:22) states that in an economy where the only certainty is uncertainty, the one source of lasting competitive advantage is knowledge. When markets shift, technologies proliferate, competitors multiply and products become obsolete, successful companies are those that consistently create new knowledge. These companies are also able to disseminate it widely throughout the organisation and quickly embody it in new technologies and products. The creation of knowledge is seen in the context of the Japanese approach that creating new knowledge is not simply a matter of processing objective information. It depends on tapping the tacit and often subjective insights, intuitions and guesses of individual employees. These insights are then made available for testing and use by the company as a whole. Making personal knowledge available to others is the central activity of the knowledge-creating company, whose sole business is continuous innovation. By this is understood that According to Nonaka the following four basic patterns for creating knowledge in an organisation exists (Figure 3):

1. Tacit to tacit.
2. Explicit to explicit.
3. Tacit to explicit.
4. Explicit to tacit.

Harari (1999) confirms this with his interpretation of knowledge when he suggests that companies should have cutting-edge skills, state-of-the-art tools, creative tools, creative freedom, business accountability for employees and speed and intelligence in everything. This is all aimed at doing something truly special that amazes customers.

Tom Davenport defines knowledge as a fluid mix of framed experience, values, contextual information and expert insight that provides a framework for evaluating and incorporating new experiences and information. It originates and is applied in the minds of knowers. In organisations, it often becomes embedded not only in documents or repositories, but also in organisational routines, processes, practices and norms.

Knowledge management (KM), as defined by the GartnerGroup, is a discipline with new processes and technologies that differentiate it from information management. New technologies are required to capture knowledge that was previously tacit. Tacit knowledge is embodied in the minds and expertise of individuals. Once captured, knowledge must be shared to leverage its value and reused in similar situations and contexts.

The unique requirements of KM have inspired many startup ventures and innovations by the information industry (Figure 9). The needs for new technologies and the technological advances have occurred simultaneously. The need is to quantify knowledge, codified in digital form in the form of documents and apply it in new situations. The codified knowledge must be found first and searching is dependent upon natural language. New KM capabilities must overcome the ambiguity and context-dependent nature of natural language.

### 3.1.2 The nature of knowledge

Table 1: Stages of technological knowledge (Bohn 1997:77)

| **Stage** | **Name** | **Comment** | **Typical form of knowledge** |
|---|---|---|---|
| 1 | Complete ignorance | | Nowhere |
| 2 | Awareness | Pure art | Tacit |
| 3 | Measure | Pre-technical | Written |
| 4 | Control of the mean | Scientific method feasible | Written and embodied in hardware |
| 5 | Process capability | Local recipe | Hardware and operating manual |
| 6 | Process characterisation | Fine-tune the process to reduce costs | Empirical equations (numerical) |
| 7 | Know why | Science | Scientific formulas and algorithms |
| 8 | Complete knowledge | Nirvana | |

Bohn (1997) identified eight stages of technological knowledge ranging from complete ignorance to complete knowledge (Table 1).

The stages that a field of endeavour goes through before it can be considered a science are (Goldratt 1990):

1. Classification
2. Correlation (The question *why* is not asked. *How* is at the centre of interest)
3. Effect-Cause-Effect (Know *why*)

The first stage, classification, in Facilities Management (FM) started about ten years ago. Numerous classifications were developed to assist in the various activities that are undertaken by facility managers. An example of this is the SAPOA standards with regards properties to calculate rentable and usable space in buildings. During the National Health Facilities Audit the concept of departments such as outpatients, operating and administrative were used to group spaces that share a common planning unit. This was used specifically to facilitate a large-scale condition and suitability analysis.

The second stage was entered about five years ago when FM developers and users realised that all the various operational and strategic FM actions need to be integrated into a holistic systems environment. The most important question at this stage is *how*. A characteristic of this stage is many different highly detailed models and approaches, but a lack of deeper scientific understanding. Both architectural briefing and design and FM are at the moment in this stage. The latter developed significantly faster than the former. The author is of the opinion that the scarcity of financial resources, energy and a general realisation that there are limits to growth (Meadows *et al*. 1975) provided the impetus. The domain of FM is far less challenging than architectural briefing and design, albeit extremely important.

The challenge of the present study is to make the quantum leap into the third stage where the *know why* becomes the characteristic. A complete mastery of *know why* would indicate that the level of science has finally been reached. Sir Isaac Newton finally turned physics into a science when he discovered the fundamental laws of motion and gravity. He asked the question why do apples fall down rather than flying in all directions? He assumed a cause for this phenomenon. He assumed the gravitational law. Explanation appears on the stage. It is a foreign word in the classification and correlation worlds where the only proof is in the pudding. Past experience is no longer the only tool. He published works such as Philosophiae Naturalis Principia Mathematica in 1687.

Nonaka (1998) clearly indicates that the classic pyramid model that consists of discrete layers of data, information, knowledge and wisdom is an oversimplification, because this model assumes a quantification of all data, hence a total transfer from tacit to implicit (Figure 2).

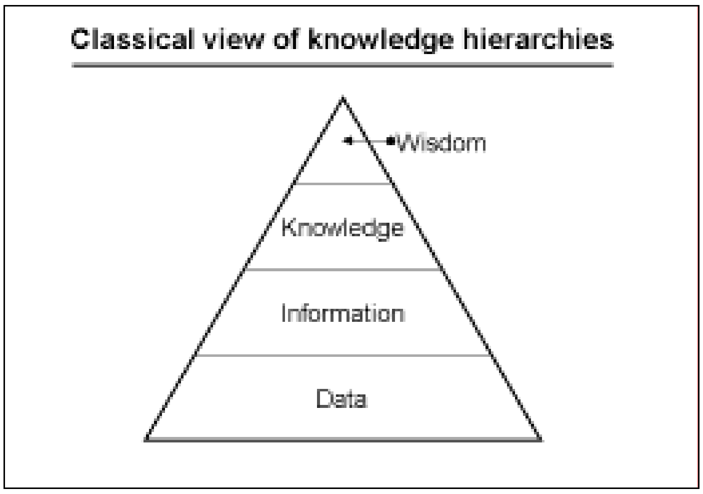


Figure 2: The classical view of knowledge hierarchies (Author)

Today it is recognised that the steps of the knowledge cycle consists of the actions of *create*, *capture*, *organise*, *access* and *use*. The most appropriate model of the knowledge cycle is by Nonaka (GartnerGroup 1998:2) and (Nonaka 1998:21). Humans have certain capabilities that lead to the four broad, fundamental human behaviours illustrated in Figure 3.

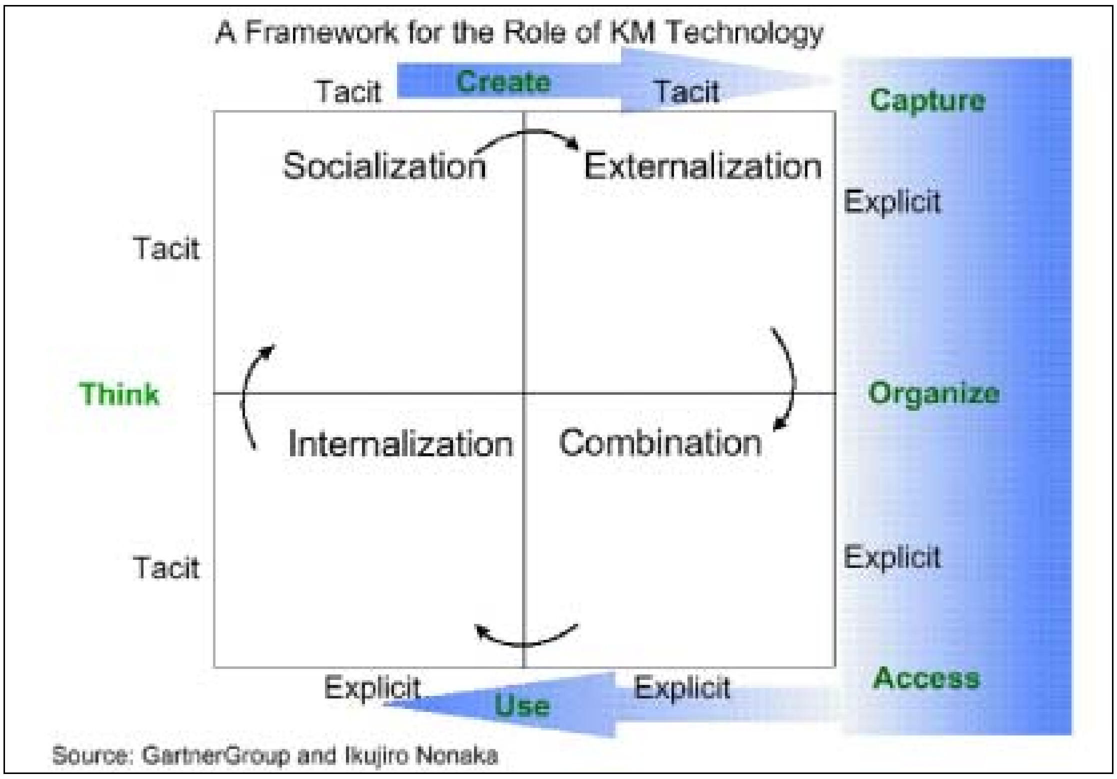


Figure 3: The knowledge cycle and process. See text. (GartnerGroup 1998:2)

### 3.1.2.1 Socialisation

This is people learning from each other by doing. Tacit knowledge is exchanged, as a by-product of collaborating, like an apprenticeship. Socialisation arises out of real-time connections among people to consolidate their knowledge. Currently, tacit-to-tacit sharing requires physical presence, but technologies such as virtual reality could eventually simulate presence.

### 3.1.2.2 Externalization

The situation where knowledgeable people consciously convert their tacit knowledge to an explicit form is called externalization. Publishing captures information, but KM requires capturing processes, conditions, rules, timing and other factors that can be re-created in subsequent situations. It requires that tacit knowledge such as a person’s experience be engineered into an explicit form. Before KM, information was captured in documents, papers, E-mail and notes. These textual sources look like long strings of words to most software. KM technology attempts to represent the traditional content and new media by exploiting natural language processing (NLP), sets of rules, document structure, context and relationships so that it can be reapplied in related situations.

### 3.1.2.3 Combination

The activity of gathering and integration of the captured knowledge of individuals and groups for access by the enterprise is called combination. Combination requires that the engineered knowledge be evaluated, transferred to other groups and communities and leveraged through reuse.

### 3.1.2.4 Internalization

Internalization is the "experience" of the explicit knowledge by individuals, who learn by making the explicit knowledge their internal knowledge. Internalization employs the techniques of pedagogy to teach knowledge that is customized and applicable to individual needs. It requires engineering knowledge to engage the attention of the user.

### 3.1.3 The current situation

Currently tacit to tacit sharing requires physical presence, but technologies such as virtual reality will eventually simulate presence. Current KM implementations depend on a small subset of the types of products that are needed for the complete knowledge cycle. GroupWare and information searches are the most used with more advanced technologies appearing in less than 15 percent of implementations. The need to externalise tacit knowledge will drive new technologies.

The ability to access stored information is far ahead of the ability to find relevant information. Turning explicit knowledge into tacit knowledge is still the purview of computer-based training (CBT) and teaching. Developments in distance learning are starting to appear from vendors such as IBM/ Lotus. The use of products for meetings, such as group decision support systems, has had very limited success (GartnerGroup 1998). Physical and virtual workspaces are still experimental and tele-technologies are today less effective than face-to-face meetings.

Improvements are expected to be made over the next five years, but the majority of sharing will continue to be through oral communication for the next 10 years or more. There are four main reasons why KM technologies will remain tools rather than replace current behaviours in the knowledge cycle:

*Situation*. Each situation that requires knowledge is unique. It is not feasible to predict exactly what knowledge is required beforehand.

*Language*. Humans share what they know through the imperfect vehicle of language. Although language is imperfect it conveys precise information within the context of use that the human brain is well adapted to. Language skills vary widely. Language is often insufficient to represent other factors that cannot be expressed verbally. This is one of the reasons why more than 50 percent of knowledge worker communication is still face-to-face, where complex things like trust can be established.

*Context*. Making knowledge explicit often leads to storage out of context. The preservation of context is very important for architectural design, because designs are always solutions within the context of numerous requirements and constraints. One of the innovative methods to store or communicate contextual knowledge is by means of business stories or novels. It is not surprising if it is considered that the oldest means of storing information is found in epics such as Homer's Iliad and the Finnish Kalevala. An outstanding modern example of this is the method that Goldratt, (1993) uses to explain the ideas which underlie the Theory of Constraints (TOC). In books such as "The Goal" he uses the context of a novel to explain manufacturing processes in the context of a manufacturing plant. The technique of using a novel is successful because it gives contextual meaning to the various powerful and innovative concepts explained. Because of the way that the human brain operates this method of externalization makes the later internalization of vast amounts of knowledge feasible.

By considering context while processing natural language it is possible to interpret the meaning of a sentence from related text by placing the individual sentence in context. According to Popov (1982) there are various levels of context that need to be considered when processing natural language.

*Textual context* is the meaning derived from the sentences preceding the current sentence.

*Situational context* is the meaning from the current sentence and is usually only given implicitly.

*Global context* is like the topic of a conversation and allows an algorithm to choose between several meanings. An example is the word overloaded that could mean having too much luggage, put to great a demand on an electrical system or the technical term in computer programming where one programming keyword such as "=" might have different meanings in different syntactical constructs.

*Local context* is the meaning derived from only the few preceding sentences. This is useful because the topic of the conversation may progress. Local context provides the most recent topic.

A simple algorithm for processing context and reference is not possible since a form of fuzzy processing is required. Researchers are experimenting with neural networks to train a computer to recognize certain common situations (frames) and to generalize about new situations.

A machine that processes natural language must be able to categorize, understand and process the wide variety of language components. Some of the different hierarchical syntactical parts of language identified by Russian analyst (Zvegintsev 1976) are:

- Discourse
- Sentences
- Phrases
- Words
- Morphemes
- Syllables
- Phonemes
- Differentiating signs

*Relevance.* The value of information is subjective. For information to be knowledge, it must get the user's attention relative to other information, comparable to a teacher-student interaction.

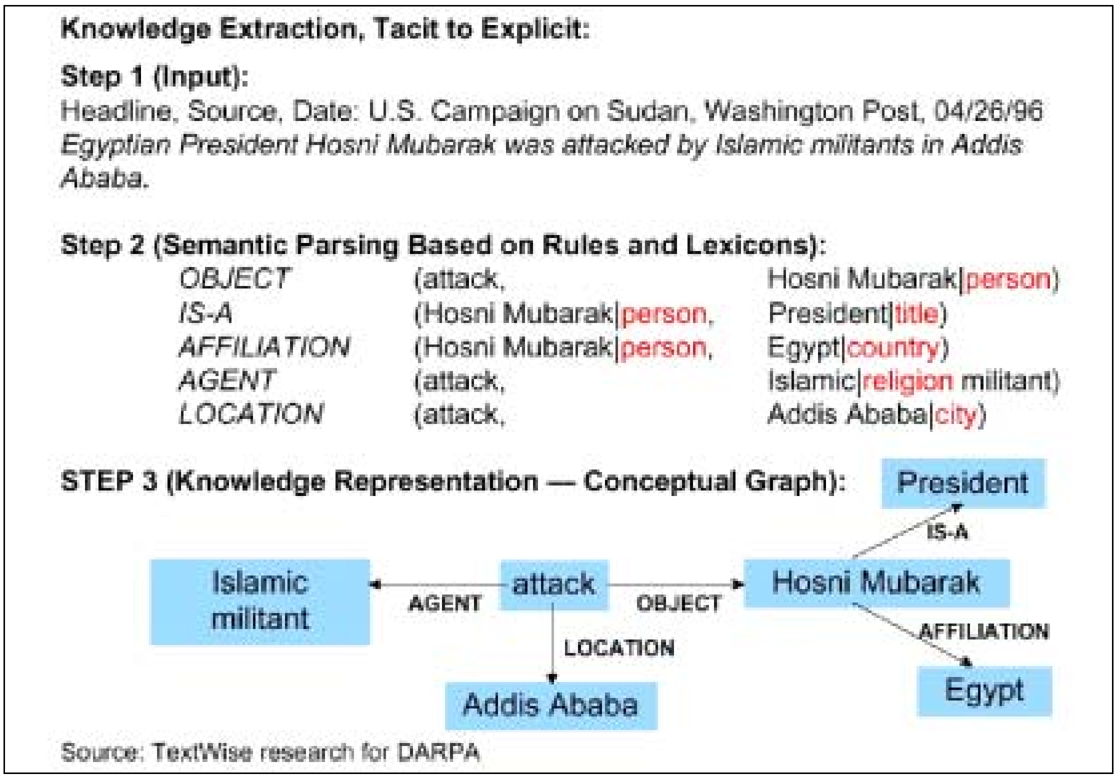
The representation of tacit knowledge is currently the focus of development, but it is unlikely that it will be used practically until after the year 2002 (GartnerGroup 1998).

#### 3.1.3.1 Desirable emerging technologies to enable knowledge management

The storage and retrieval of knowledge in written form was viable with paper and limited online access before the advent of the World Wide Web. The present unprecedented quantities of information online made the retrieval of documents or records inadequate. New capabilities are addressing the conversion of text into more usable forms. It is now important to achieve summarization in order to find relevant knowledge. This is a capability that would increase the quality of stored knowledge. However, it depends on accurately representing the meaning of text. With the current information overload, users will want to know what a document is about rather than having to read it or a summary. Capabilities are being developed that describe the people, places and things discussed in document. This is at a far higher level than a pure keyword search. Users can search in a natural language way for subjects such as "Tell me about Microsoft, Internet components and Visual Basic". Unlike present search technology that search for those words specifically, representation technology will identify what Internet components are and respond with the latest Microsoft Internet Information Server technology used. The barriers to computer understanding of language results from the innumerable ways to express the same thing, the different meanings of words and the inherent imprecise nature of language structure. If information technology fails to achieve significant breakthroughs in representation technology soon the knowledge community will be faced with vast but virtually unusable knowledge stores.

If intelligent processing of design information is to take place on the electronic desktop then relevant structured information needs to be delivered to the electronic design tools. This indicates the necessity for a flexible and self-describing design language.

Concept extraction is at the same time one of the most promising and problematic of the emerging capabilities. It depends on NLP to parse sentences according to rules (Figure 4). In this particular example the Xerox Inxight Hyperbolic Tree was used in the Syracuse University TextWise system. *President* is capitalized and directly precedes a name. The lexicon says *President* is a title and *Mubarak* is a name. In a newspaper article users want to know how concepts are used and not just the fact that they occur. In a document that contains the concept “Chrysler” it could be an advertisement, stories about the company’s stock performance or an article about a recent merger.



**Knowledge Extraction, Tacit to Explicit:**

**Step 1 (Input):**

Headline, Source, Date: U.S. Campaign on Sudan, Washington Post, 04/26/96
*Egyptian President Hosni Mubarak was attacked by Islamic militants in Addis Ababa.*

**Step 2 (Semantic Parsing Based on Rules and Lexicons):**

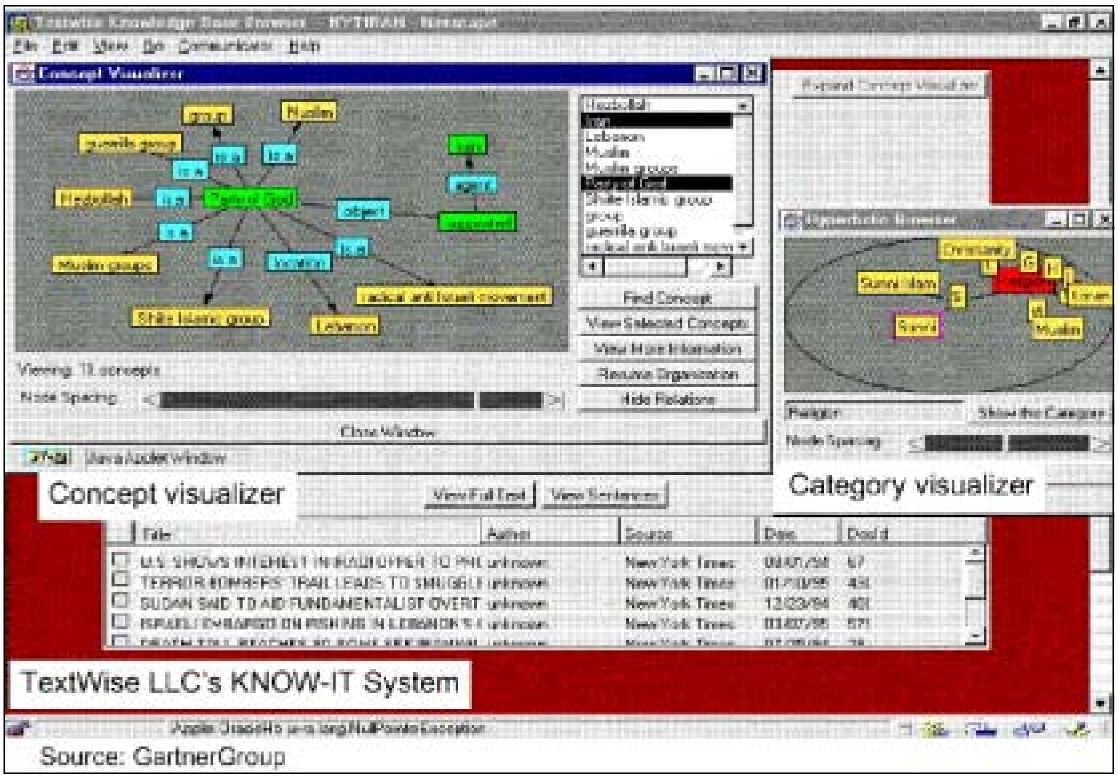
| | | |
|---|---|---|
| *OBJECT* | (attack, | Hosni Mubarak\|person) |
| *IS-A* | (Hosni Mubarak\|person, | President\|title) |
| *AFFILIATION* | (Hosni Mubarak\|person, | Egypt\|country) |
| *AGENT* | (attack, | Islamic\|religion militant) |
| *LOCATION* | (attack, | Addis Ababa\|city) |

**STEP 3 (Knowledge Representation — Conceptual Graph):**

Source: TextWise research for DARPA

Figure 4: Concept extraction for representation (GartnerGroup 1998:8)

By representing the concepts according to conceptual relations, the technology could infer the difference. Once concepts are extracted, they can be stored in databases according to their usage and relationships Concepts are also the basis of visualization capabilities. They are displayed on a screen where the relative proximity of each concept reflects relative similarity of meaning. This provides the user with a conceptual map of a knowledge domain, enabling navigation to the units of stored knowledge. It also shows relationships such as affiliation. For the next two years automatic linguistic representations will require access to manual checking through visual user interfaces to expose inaccuracies.



| | Title | Author | Source | Date | Doc# |
|---|---|---|---|---|---|
| ☐ | U.S. SHOWS INTEREST IN IRAQI OFFER TO PHI | unknown | New York Times | 03/07/98 | 67 |
| ☐ | TERROR BOMBERS' TRAIL LEADS TO SMUGGLI | unknown | New York Times | 01/10/95 | 431 |
| ☐ | SUDAN SAID TO AID FUNDAMENTALIST OVERT | unknown | New York Times | 12/23/94 | 401 |
| ☐ | ISRAELI EMBARGO ON FISHING IN LEBANON'S | unknown | New York Times | 03/07/95 | 571 |
| ☐ | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |

Figure 5: Visualisation of representation (GartnerGroup 1998:9)

Knowledge representation will expand beyond linguistics to include technologies that will converge over the next five years. Significant developments in collaborative filtering resulted in successful products such as grapeVine, NetPerceptions, Firefly and WiseWire. Users with similar profiles are considered sources of recommendations to each other. Profiles use representation technology to capture similar interests.

Concept extraction provides a way to describe documents. The concepts can be used as attributes in a database. Expert systems continue to be focused on narrow domains, which can be described by rules.

Knowledge representation is the conversion of captured knowledge into a reusable form. But it focuses on the tacit-to-explicit part of the cycle. It must be integrated with sharing media to overcome the scalability issues of space and time. Sharing over time has both a historical and a coordination dimension. Knowledge must be maintained over time or it will lose its value. It must be current, as is true of any information. Knowledge workers try to interact synchronously, in real time, because they want the latest knowledge applied to the current situation. Scheduling a simultaneous interaction is increasingly difficult, especially with worldwide enterprises. The two notions of time exacerbate each other. One of the largest demands in KM is to overcome the need to interact in real time to share current knowledge. The reusability test requires that person *A* must be able to make sense of and apply person *B*'s knowledge at person *A*'s time and place. When this is not feasible, users must resort to sharing media by means of E-mail and teleconferencing.

### 3.1.4 Knowledge management architectures

There are several barriers to a next-generation responsive information system, which can be rapidly composed or adapt itself to a new environment. Presently vendors are reluctant to support truly open architectures. Neither the vendors nor the users understand the importance or the nature of the interface between modules. In any modular system, the interface definition is critical to the assembly of those modules into larger functional modules. Many vendors claim to have open architectures, but they are only open if one adheres to their proprietary standards. An architecture or framework can really only be called open if it is freely available and the command interfaces and data interfaces are widely published and easily accessible. The World Wide Web fits this picture.

Another major barrier to the imperative is the multiplicity of standards for specific domains and the length of time it takes to create a new standard. While the availability of certain standards will enable the move to information infrastructures, there are many conflicts and many standards are lacking. There are multiple overlapping standards in high-level communications, particularly for passing of messages among distributed objects on heterogeneous systems and for data exchange (Table 2). There are also diverse standards for graphical user interfaces for documents, images, video and sound. These are all important for next-generation manufacturers. The multiplicity of standards makes access and use of the data difficult. This particularly true in the design world where AEDES and ARGOS needs to be implemented. STEP has come a long way in addressing these problems but it is taking too long and has become so complex that users are adapting other standards to allow them to progress.

Table 2: Current and emerging multiple standards that form a barrier to responsive NGM information systems (NGM 1997)

| **Topic** | **Subtopic** | **Example available or emerging standard** |
|---|---|---|
| Communication | | CORBA, http, OLE/COM/DCOM, DCE, ISO/OSI |
| Data exchange | Product | STEP and its various APs |
| | Geometry | STEP AP 210, DXF, STL, HPGL, ACIS SAT |
| | Text, hypertext | http, SGML, XML, RTF |
| | Images | JPEG, BMP, GIF, TIFF, DIB, PCX, MSP |
| | Simulation data | SAVE MDF/ CDF |
| | Production plans | ALPS, STEP AP |
| Presentation/ GUI | | Windows, X-Windows, Open GL, Tcl/Tk |
| Process modelling | | IDEF, NIAM |
| Computer languages | | C, C++, ADA, Java, Visual Basic |
| Sound | | WAV, AU, RA, MIDI, RMI, AIF |
| Video | | MPEG, AVI, MOV |
| Database Query Languages | | SQL, SDAI |

If there were standards for every aspect of the command and data interfaces among modules, this would not be a barrier. Lacking these, it currently takes too long for a group of companies to agree on even the product data exchange standards let alone all the other standards that must be considered. For example the Distributed Computing Environment Group (DCE), the Object Management Group (OMG) and Microsoft are all working on protocols for message passing between distributed objects on heterogeneous systems. The HTTP Working Group is working on similar standards. The contributors to the NGM (Agility Forum 1997) project are of the opinion that CORBA, OLE and DCE will merge with http and Java having a strong influence on all. Java and http will have a strong influence on the whole communication domain, but it is not yet clear what it will be.

During a recent investigation by the author of 16 companies that positioned themselves in the KM domain it became clear that the IT industry has already done a tremendous amount to gain a deeper understanding of the exact requirements and architecture of a Knowledge Management System (KMS). The solutions investigated ranged from attempts to integrate the multimedia enterprise knowledge in various forms in a readily accessible user interface to intelligent cross-linked repositories with highly configurable knowledge profiling environments.

All the local suppliers are unanimous in their opinion that it is at this stage almost impossible to fully quantify knowledge that originates at the tacit level. Tacit knowledge is seen as the experiential knowledge of people that exists mostly in the minds of professionals. For this reason all the solution providers are concentrating on the communications and access profile aspects of KM.

The main technical requirements to implement a KM system successfully can be summarised in Table 3. In order to implement a comprehensive briefing and design system the infrastructure needs are very similar to the requirements for KM. The only difference that can be identified is the type of knowledge that is stored and the mix of software tools that would be used to solve the various problems in the design stages.

Table 3: The main requirements for a Knowledge Management enabling environment (Collated by author)

| Main Requirements | Expanded software requirements for integrated knowledge based architectural briefing and design system |
|---|---|
| **Communication** | **Network infrastructure**<br>• Flexible high speed network configuration supporting distributed objects and an ubiquitous service environment.<br>**General office knowledge content management**<br>• Store all forms of project and office knowledge such as scanned paper documents, electronic word processing documents, video and voice recordings into an information base.<br>• Provide support listing, browsing, sorting, grouping, filtering and searching of the knowledge base.<br>**Teams that collaborate in real time and over distance**<br>• Conversation services with transcript functionality for distance discussions.<br>• Video conferencing for virtual meetings.<br>• Screen sharing services for sharing of the document creation process, virtual white boards and application sharing.<br>• Streaming media services for recording virtual meetings and video (meeting) on demand.<br>• Event and meeting databases for organising and optimising meetings.<br>• Home pages on Web servers for each task community, team or expert to speed up the access to project information (knowledge sources) |
| **Design team flexibility and responsiveness** | **Team skills profiles**<br>• Directory and membership services that support the building of communities through grouping people together into expert teams working on the same set of information.<br>• Forum services to create workspaces for communities and teams that contain all interest-related data.<br>• Self-subscription services to specific matters of interest for dependent information delivery and subscribing.<br>• Organisation databases integration like people skills and human resource databases for enhancing community, team and experts information and searches this information.<br>**Responsibility and accountability**<br>• Services to assign responsibilities to members of the team. |

| | |
|---|---|
| | • Workflow services for automatic trace ability processes based on roles and Subject Matter Experts (SME).<br>• Tracking services that follow team contacts and team activities.<br>• E-mail services for automating notification, routing and simple workflow services. |
| **User Interface and information search** | **User interface**<br><br>• Personalization systems that allow customisation of the computer project interface.<br>• Web browsers with the ability to include e-mail, project data and design intelligence tools for easy access.<br><br>**Ubiquitous demand driven design and construction information**<br><br>• Construction industry catalogue and search services.<br>• Availability of material and product databases.<br>• Acquire pre-packaged starter kits from other companies via the web.<br>• Services to build own office electronic storage that combines often used information from external origin.<br>• Notification services that react on changes in design or fundamental information contained in local catalogues and integrate with the e-mail system.<br><br>**Reporting**<br><br>• Dynamic project reports available on the Internet or intranet. (OLAP technologies and services)<br>• Configurable ad-hoc, multi-media query builders that can be profiled to serve the needs of the user or the task at hand. |
| **Project resource integration and access** | **Information access**<br><br>• One convenient entry/ access point to all project information and applications.<br>• Knowledge indexing services that can index all the documents for easy subsequent retrieval.<br>• Subscriber services where the service provider maintains the infrastructure and the subscriber accesses the service by means of his Internet Explorer/ Netscape driven by credit card payments, if and when required. This saves the user buying less often used specialised tools to analyse energy and cost if and when required.<br>• Rapid design concept selection for new designs and projects based on previously stored structured knowledge.<br><br>**Life cycle continuity of information**<br><br>• Building design and operational information that is maintained over the life cycle of the building.<br>• Captures project briefing and design decisions and knowledge automatically and during the course of the project in the form of plug and play design starter kits. |

#### 3.1.4.1 Hypertext based systems

One of the most basic means to organise electronic knowledge is by means of hypertext driven systems. An example of this is the use of a World-wide Web page with hyperlinks that could point to documents such as word-processing, spreadsheets, images and presentation documents. The capabilities of the Microsoft Windows operating system facilitate in-place activation.

The main advantage of hypertext is that the reader can choose his own associative way through hypertext, depending on his background knowledge, interests, context of use and his task at hand. One of the difficulties for the reader of hypertext is to get an overview of the structure of the knowledge presented. To this end hypertext systems normally provide tours.

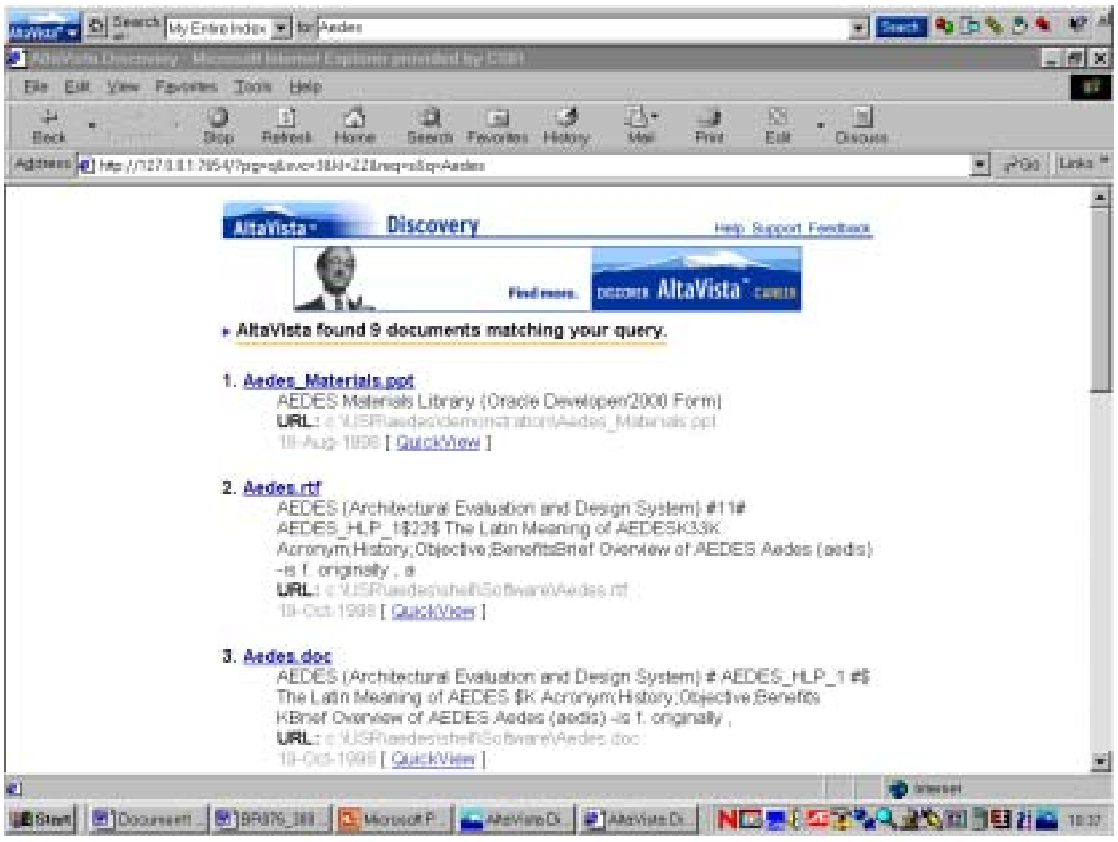
History lists provide intelligent backtracking. Bookmarks also assist to orientate the user in the vast hypertext landscape.

In *AEDES* a context sensitive help file was created that tested some of the underlying principles mentioned above. The present help file was created in the Microsoft .HLP format. This format will very likely be superseded with a HTML, www format help file. At the moment the .HLP format is predominant on the desktop, but the gradual move to HTML format is becoming evident in the Microsoft Office 2000 suite.

### 3.1.4.2 Search engines such as Alta Vista Discovery

The downloadable Alta Vista Discovery search engine provides a convenient means to index documents on the personal computer and in a network environment. The author tested the software and found it a good solution. The only problem is that the index needs to be rebuild from time to time to keep it current. This action takes a long time, typically an hour on the present equipment that we are operating on. The look and feel of the output is exactly the same as the Alta Vista search engine used in the Internet environment (Figure 6). In the example the search word was “Aedes”. The response was a list of all documents that contain the word “Aedes”. In the example Powerpoint, rich text file and Microsoft Word formats were found along with the local directory paths. The search engine is also capable to index email messages. A very rudimentary summarise facility is provided, but unfortunately not of much use, because it purely extracts random fragments of text up to the specified number of words.

The main advantage of this environment is that it can exist independently of any other system software. It provides good search and find facilities for a diverse range of documents. It can almost be seen as a very basic document management system.


Search My Entire Index for Aedes
Search
File Edit View Favorites Tools Help
Back Stop Refresh Home Search Favorites History Mail Print Edit Discuss
Address
Go Links
AltaVista Discovery
Help Support Feedback
Find more.
Discover AltaVista Career
Start Document1 BR876_38 Microsoft P AltaVista Di AltaVista Di
Internet
10:37

AltaVista found 9 documents matching your query.

1. Aedes_Materials.ppt
AEDES Materials Library (Oracle Developer/2000 Form)
URL: c:\USR\aedes\demonstration\Aedes_Materials.ppt
18-Aug-1998 [ QuickView ]

2. Aedes.rtf
AEDES (Architectural Evaluation and Design System) #11# AEDES_HLP_1$22$ The Latin Meaning of AEDESK33K Acronym;History;Objective;BenefitsBrief Overview of AEDES Aedes (aedis) -is f. originally , a
URL: c:\USR\aedes\shell\Software\Aedes.rtf
19-Oct-1998 [ QuickView ]

3. Aedes.doc
AEDES (Architectural Evaluation and Design System) # AEDES_HLP_1 #$ The Latin Meaning of AEDES $K Acronym;History;Objective;Benefits KBrief Overview of AEDES Aedes (aedis) -is f. originally ,
URL: c:\USR\aedes\shell\Software\Aedes.doc
19-Oct-1998 [ QuickView ]

Figure 6: A typical Alta Vista Discovery screen (Author)

### 3.1.4.3 Essential elements of a knowledge management architecture

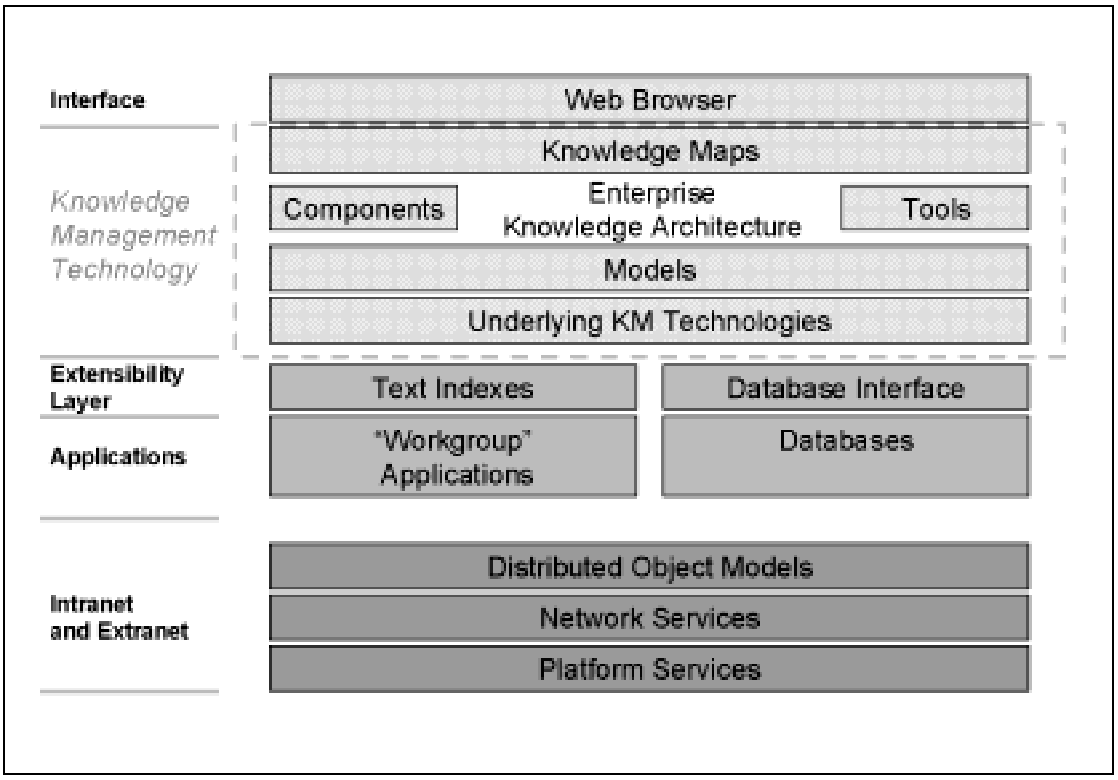


Figure 7: Knowledge management architecture (GartnerGroup 1998:3)

Figure 7 details the technologies that are targeted at KM. They are classified according to the architectural fit and the KM process that they support. The enterprise infrastructure is a networked service supporting distributed object models such as the OMG CORBA and Microsoft DCOM. Directory services and security services are becoming openly accessible by the upper levels of software. Workgroup applications continue to be silos of unstructured, mostly textual information that is set apart from database silos of structured information. Progress is being made to bridge the text data wall through SQL. However currently KM must be satisfied with the integrated retrieval of records. Text indices are easily offered using the centralised model of the Internet, but databases demand complex program interfaces and additional structures such as data warehouses.

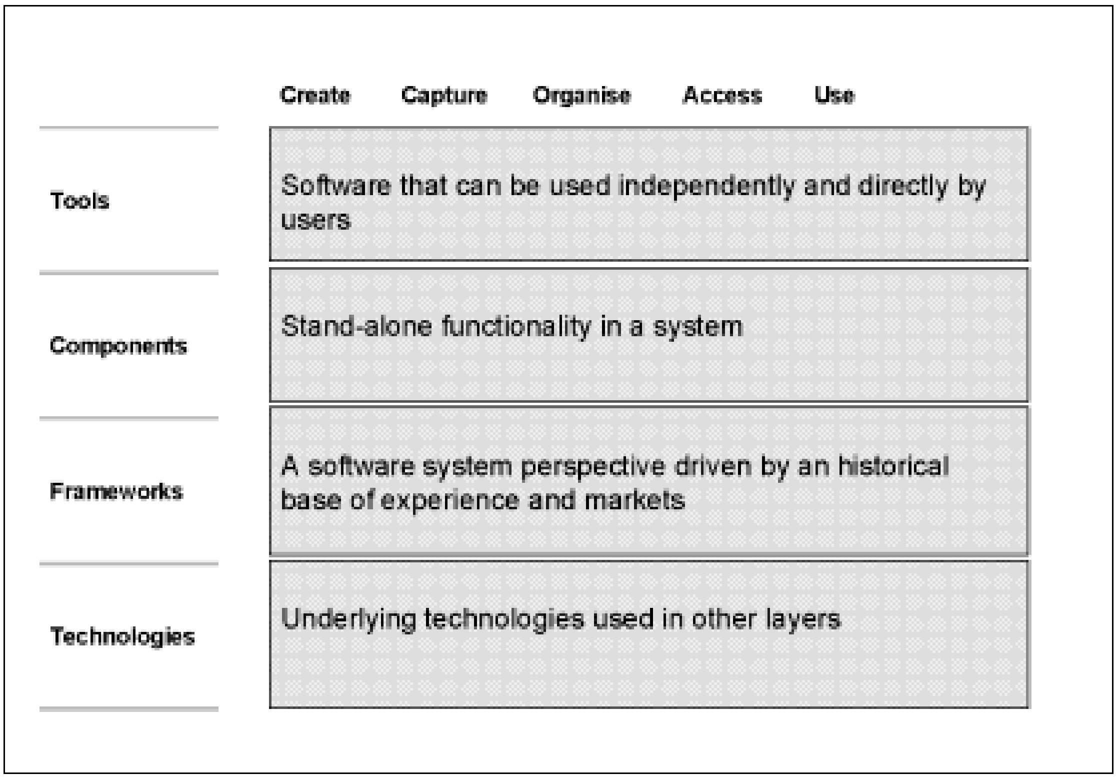

Create
Capture
Organise
Access
Use
Tools
Software that can be used independently and directly by users
Components
Stand-alone functionality in a system
Frameworks
A software system perspective driven by an historical base of experience and markets
Technologies
Underlying technologies used in other layers

Figure 8: Knowledge Management technology model (GartnerGroup 1998:4)

The KM technology layer of the architecture defines four essential elements (Figure 8). *Technologies* include mathematical algorithms, statistical techniques and database architectures. *Frameworks* define the central focus of an architecture, reflecting many subtle differences in vendor orientation. Information retrieval vendors are likely to be document-centric and more experienced with unstructured content. Database vendors are likely to be data-centric and experienced in high-speed transaction processing. *Components* are units of functionality that can be plugged into other products, but are not directly used by the knowledge worker. Tools are directly manipulated by the end user and are often branded. Good examples of desktop productivity tools are Visio, Word and Powerpoint. The software available today provides different levels of support for each of the five process steps. However *neural networks*, *Baysian Nets* and *linguistics-natural language processing* are all unproven. *Neural networks*, a development of artificial intelligence, learns to recognise patterns. *Baysian Nets* is a probabilistic model from past to future events. *Linguistics-natural language processing* is the representation of everyday language for computer understanding.

New deployment of advanced, underlying technologies presents a conundrum. The technologies are unproven but demanded by the market. *Natural language processing* (NLP) development and *neural network* application are both being funded by the same agencies such as the U.S. Department of Defence's DARPA (High Performance Knowledge Base Program. Neural nets require training that makes it very difficult to succeed with single pass queries. NLP is still the most promising for capturing the meaning of language, a capability that will soon be essential for KM.

### 3.1.5 Microsoft's approach to KM

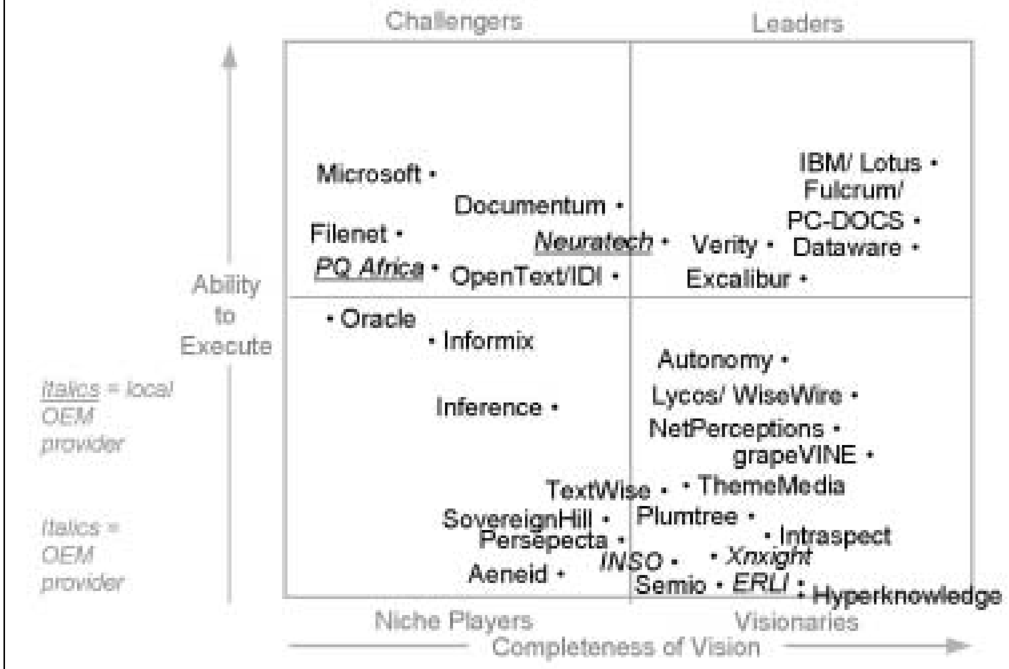


Figure 9: Knowledge Management vendors (Based on GartnerGroup 1998:16)

Although Microsoft remains strongly tied to the desktop, recent innovations in enterprise OS and network technology indicate that they are in the process of functional leapfrog. Figure 9 indicates the current position of Microsoft in comparison to other software developers with regard KM software. However in the present study the slight lack of vision as indicated is more than adequately compensated by the excellent desktop, document centric, convenient and  portable object technology. Microsoft currently offers a comprehensive platform, albeit slightly conservative in terms of what a fully functional KM system will require. To ensure the success of the KMS very special professionals will be required such as Knowledge Architect (KA) and chief Knowledge Officer (CKO).

The Microsoft approach to KM is not unlike the generic structures illustrated in Figure 7. They are also of the opinion that the two prerequisite technologies for all KM systems are a *Complete Intranet* and *Messaging and Collaboration* (Leibmann 1999).

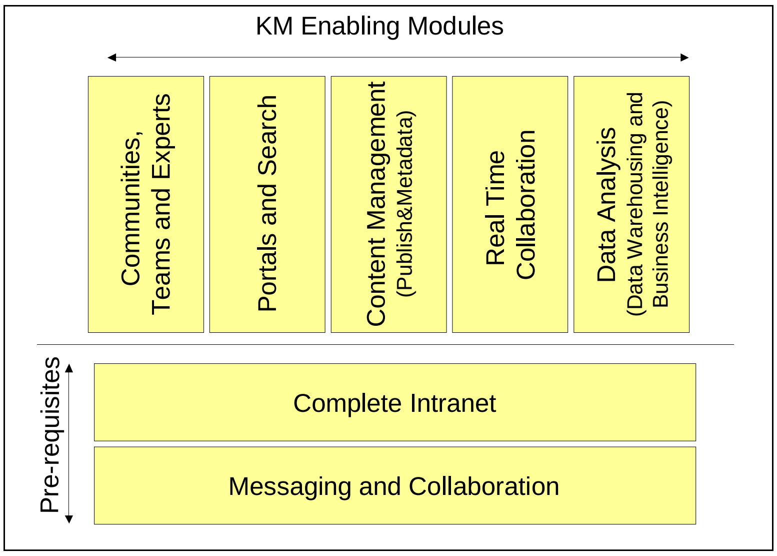

KM Enabling Modules
Communities, Teams and Experts
Portals and Search
Content Management (Publish&Metadata)
Real Time Collaboration
Data Analysis (Data Warehousing and Business Intelligence)
Pre-requisites
Complete Intranet
Messaging and Collaboration

Figure 10: The modules of a Knowledge Management evolution (Leibmann 1999:7)

The remaining KM-Enabling Modules extend the basic infrastructure to a KM system that includes services like *Content Management*, *Information Delivery* and *Data Analysis*. Automated services such as *Data Tracking* and *Workflow* processes are also included as part of the *Community and Team* competencies.

The KM-Enabling Modules have a modular character. Although some of the modules profit from the implementation of a previous module, they can be chosen in any order related to the specific business case that needs to be accomplished. Real-time Collaboration services, such as video conferencing, can be included on top of the pre-requisite technologies, but are enhanced by the meta data services provided in the *Content Management Module*. Meta data is a special database of where diverse sources of data can be found.

### 3.1.5.1 Messaging and collaboration

IT systems intended to support KM need to support the capturing of undocumented information such as human thoughts, sharing of ideas and documents. It is important to find this information efficiently. Another prerequisite of an IT system that supports KM is the existence of a set of common tools that are well known by all knowledge workers.

The tool that is used to provide an entry point to this IT system presents the information and controls all interaction with it. It needs to be capable of handling all the information that is part of the working environment of the knowledge worker. Ideally only one tool or application should exist for this interface.

The entry point to the information and applications in a KM system is also called a “Portal Service”. If the same environment also supports also the creation of content, it is called a KM desktop. The capabilities of web browsers make them ideal candidates for this task.

### 3.1.5.2 Complete Intranet

An information network that provides access to all the data needed supports this module of KM. Decisions must be taken fast enough to get a competitive advantage. A Complete Intranet KM system should enable people to find the right information or sources for helping solve problems or drive decisions.

### 3.1.5.3 Communities, teams and experts

The two pre-requisite technologies of KM put all collaboration and document-based knowledge sources together enabling the knowledge worker to browse information objects based on knowledge groups. *Communities, teams and experts* add the next level of sharing knowledge and turning it to results.

Teams differ from communities in that teams are task driven and communities are interest driven. A team usually works closely together, in a workgroup, on the same tasks and goals. In many cases the information produced by a team is closely held within the team until it has reached a level of completeness where it can be shared, for example in a review, with a broader audience. Communities are mostly driven by interests in the same area and are more loosely coupled, for example by subscriptions. Communities are especially useful for building knowledge to higher levels, often by getting successive levels of input from a wide audience.

The role of an expert is to qualify and filter information. Often an expert is related to a limited set of subjects. Subject matter experts (SME) can be defined in two ways. He is either an organisational function (defined by the KA) or as very knowledgeable person who is a well-known expert in his team or organisation, assuming the status of an SME for contributing high quality information or for reviewing it.

The SME is an important role for a KM-related Information Web or Intranet. In traditional Intranet solutions, there is little control over who can store or upload information into the Intranet. This is not a bad thing and is desirable in order to build an extensive information repository. To maximise the usefulness of the Intranet, the information should be filtered, classified and grouped. This process is part of the responsibility of the SME.

*Communities, teams, and experts* are also used for the controlled process of putting information into the KM system. Filtering, qualification, approval or more complex workflow processes for documents and other electronic data need to be established. In a KM system these processes are not strictly based on traditional organisational roles such as manager, reviewer, approver and author but more on Subject Matter Experts. This can add a great level of dynamic and flexibility to the KM system and the automated processes.

### 3.1.5.4 Portals and search

Portal Services like Yahoo, Lycos and Excite are well known. They allow consumer oriented services such as easy information shopping. Those Portals categorise personal interests in groups like “News”, “Sports”, “Economy”, “Education”, “Science” and “Entertainment”. They allow for easy browsing within those groups in building a logical hierarchy of subgroups or forums. The browsing and search services support the consumer in his quest to gain knowledge out of the large Internet information store. Another benefit of these consumer Portals is the high customisation provided for its visitors. Objects of interest can be bookmarked in a personalised Portal, allowing for immediately access when revisiting the portal site.

This technique when applied to business-oriented goals is one of the key KM-enabling modules following the same idea of the consumer-oriented portals in the corporate world. Business Portals provide the knowledge workers within the company, and also external suppliers and customers, with instantly task-relevant information objects. A Primary goal of a portal is the transparent enterprise, hiding the complexity to access knowledge stores. Even if the user accesses legacy information stores he should not be aware of it.

Examples of Business Portal Information Objects are:

- Corporate and team links.
- Team application links.
- Incoming mail notification and headers.
- Personal tasks.
- Corporate search.
- Integration of business intelligence data.

From the examples, some direct organisational tasks can be derived. Teams in the enterprise need these definitions in order to locate internal or external company information. This allows them to successfully include links to that information into the portal.

This Module also defines the creation of catalogues that build groups of related information based on business needs over structured and unstructured enterprise information (KM information base) to allow for full-text search against the partitioned data. An extension to the Catalogues is the definition of Searches against these Catalogues.

In order to define the Catalogues for an organisation, there has to be a very good understanding of the business and its processes. At this stage, the Knowledge Architect needs the support from the different divisions, business units and departments that understand how their information is organised and is related to their business goals, tasks, and needs.

### 3.1.5.5 Content management

Portals and Search address the problem of searching knowledge using all information sources in the enterprise such as structured and unstructured internal information objects. Examples of these are office documents and collaborative data. External sources such as partners, suppliers and competitors can be identified. Other external sources such as the Internet provide a tremendous potential for knowledge if the criteria for including such information are well chosen.

All the pools of information sources that are part of and accessible to the KM system combine to build the KM information base. This Module handles how knowledge assets get into the KM information base. To handle this new complexity of the KM information base and help knowledge workers to stay focused on solving business problems without disappearing in technology a sophisticated KM taxonomy needs to be built based on meta data. It also needs to publish information in the knowledge base. The KM information base must then be made accessible through operations driven by the meta data complex.

When publishing, several things should be considered concerning the KM taxonomy. Meta data tagging of documents is important for the quality of the content in the stage of document publishing. However it should not be a burden for people to submit information. The KM system must encourage users to submit information. Positive aspects for promoting this condition are the building of well-focused Communities in order that users feel part of and respected in a concentrated team and do not loose their inclination or motivation to submit. Building huge submission and posting systems where users do not get recognised or rewarded will discourage them from providing their knowledge, which will prevent the company from evolving a culture for knowledge management.

### 3.1.5.6 Real-time collaboration

The knowledge on a specific subject is often not in documented form and is therefore lost to the organisation. There are ways retrieve the lost knowledge into a state where an IT system can manage it. This especially focuses on areas where computers can be used to exchange thoughts, documents and other aids for capturing such tacit knowledge for the KM information base.

The process of capturing tacit knowledge can start with the introduction of simple computer-based chat services. Regular meetings arranged with expert groups to talk about specific topics can be extended with these services, well known from the Internet and enriched by building automatic transcripts for the chat sessions. Transcripts can be enriched with the corporate KM meta data and stored in the KM information base for later search and retrieval.

More advanced services like video conferencing follow the same concept. The video stream is recorded on video equipment and is subsequently transferred to the KM system. Descriptions and meta data are either merged with this video stream or can be stored in parallel on a file or database. For cultures where such virtual meetings are common, an event database is typically built where upcoming and past meetings are stored, together with event titles and descriptions. They can be listed or searched by subject matter by means of meta data. A hyperlink is provided so those users may join a virtual meeting. If the meeting takes place in the future, integration into the e-mail system ensures that this event is marked in the calendar, and on the event date a reminder automatically guides the participant to the virtual meeting. After the event or meeting, on-demand services will make that knowledge available, by providing the recorded video out of the KM information base to the KM desktop.

When integrating this technology into the automated KM services scenario, notifications are sent automatically to the appropriate knowledge workers to remind them of an interesting meeting or event. The appropriate URLs can also be listed on the KM portal.

A hybrid of abovementioned technologies is the integration of presentation techniques. In this an online presentation that consists of slides is sent over the network. The audience receives the video, audio and slides of the presentation on the KM desktop. The chat service is integrated as a separate area on the KM desktop and enables the audience to type questions during the meeting into the chat area. These questions are transferred to the presenter or a person controlling the online presentation. On receiving the questions the presenter can answer them during or at the end of the event. The slides as a document, the chats as a transcript document and the audio and video as a stream are linked together and stored in the KM system.

The same technologies not only make virtual events available for the KM information base, but also real events like conferences. Each session on a conference can be recorded and than be made available for all employees in the events system on the corporate net. Another solution is to produce CDs of the sessions and distribute them to all subsidiaries or make them orderable for interested employees.

Real-time Collaboration KM also provides support for sharing the creation process. It enables distant knowledge workers to share a single virtual working space and to collaborate on the creation of documents. This includes not only the sharing of the creation process using the productivity suite, but also white board functionality. This kind of technology is also known as screen sharing.

## 3.2 Knowledge based design

### 3.2.1 Introduction

This section investigates the various Knowledge-Based Design approaches in an attempt to see whether they could add value to the final solution proposed. Over the years many different approaches were used that had different levels of success. It is intuitively sensed that highly structural and prescriptive methods are not suitable. It also appears that certain information in design is well defined whereas other information is incompletely specified and vague. This section analyses Artificial intelligence and design, problem-solving architectures and Case-based design.

The first generation of Knowledge-based Design Systems (KBDS) was characterised by the dominance of logic models and rule-based systems then prevailing within expert systems technology. The paradigm of Knowledge Engineering (KE) appeared to be promising and relevant to design. KE turned out to be far more applicable to Knowledge Management (KM) that is likely to form the holistic operational framework for globally enabled design and project environments. KE has limited use for the range and complexity of design tasks. Debenham (1998:1) states that a unified KE methodology treats data, information and knowledge in a homogeneous manner. However, with a few exceptions, models of expert knowledge appeared to have limited utility for the range and complexity of design tasks (Oxman *et al.* 1994).

Debenham (1998:23) defines a Knowledge-based system as a system that represents an application containing a significant amount of real knowledge and has been designed, implemented and possibly maintained with due regard for the structure of the data, information and knowledge. A significant amount means that the application boundary of the system should identify an area of the application that is appropriately dealt with using knowledge-based systems design techniques.

In an application:

- Data is the set of fundamental, indivisible things (Debenham 1998:18).
- Information is the set of implicit associations between data things (Debenham 1998:20).
- Knowledge is the set of explicit associations between the information things and/or the data things (Debenham 1998:20).

An expert system is a system in which knowledge is represented as it is, possibly in the same form that it was extracted from an expert. In an expert system the represented knowledge should endeavour to solve problems in the same way as the expert knowledge source solved them.

Debenham (1998:23) identifies differences between Knowledge-based systems and expert systems:

- Expert systems perform in the manner of a particular trained expert. A knowledge-based system is not constrained in this way. In a knowledge-based system the represented knowledge should be “modular” in the sense that it can easily be placed alongside knowledge extracted from another source.
- Expert systems do not necessarily interact with corporate databases. In general, knowledge-based systems belong on the corporate system platform and should be integrated with all principal, corporate resources.

Carrara *et al.* (1994) states that computer-aided architectural design research has inherited the unanswered questions first raised by theorists like Rittel, Simon and Schon. To this has been added the additional complexity of representing the answers in an explicit and complete way so that they can be handed over to and reproduced by machines. The combined search for solutions to these questions is called Knowledge Based Computer-aided Architectural Design (KBCAAD). This title implies that the search for tools that could assist designers in the design of buildings relies on one hand on understanding the cognitive processes of architectural design itself as well as the theories, methods and techniques that have been developed outside the discipline of architecture. It is the synthesis of these two sources that holds the promise that an appropriate balance will finally be found.

Architectural CAD researchers have been focussing their attention on the cognitive aspects of the architectural design process since approximately 1990. They have been constructing models of design knowledge and reasoning. They developed data structures to represent them computationally. Although the models are unique to the discipline of architecture they were borrowed and adapted from other disciplines such as Artificial Intelligence (AI) and product development. Inference engines that were prominent at the height of the interest in expert systems have not proved themselves for design applications of substance.

Due to the complexity of design, systems for design have often defined the task with artificial narrowness (Hinrichs 1991:3). In AI, as in Fuzzy Set theory, progress in the past was made by limiting the universe of discourse or even closing it in an attempt to simplify the enormously complex design problems. To make the systems tractable the following typical four approaches were used (Hinrichs 1991:3):

- *Selection.* Select components to instantiate a skeletal design. Selection problems are typically constraint satisfaction problems in which all variables to be satisfied for are known ahead of time. The space of possible components is given as if from a catalogue.
- *Configuration.* Arrange a given set of components. Configuration is essentially the dual of selection. This method concentrates on the relationship between components rather than the components themselves.
- *Parametric.* Fix numeric parameters. Parametric problems are similar to selection problems except that the components are quantities and the task is usually to optimise or to partially satisfy constraints.
- *Constructive.* Build up designs from components. Constructive design is analogous to planning, in that components take on the role of operators or actions. Typically, the space of components is fixed throughout the design process.

Hinrichs (1991:3) observes the fact that if design problems are viewed as instances of abovementioned types, they can often be solved using efficient algorithms and heuristics. However, rigid classifications do not capture the flexibility that real designers exhibit. A model of design or a system that supports design in an open world should be able to use any of the four generic types of design.

In addition to the different types of design approaches, AI research has explored different approaches to the process of design. Hinrichs (1991:3) summarises some of these approaches as:

- *Pure synthesis*: Construct designs from the bottom up. The pure synthesis approach assumes that the design problem space is basically a very large graph. If the appropriate heuristics can be found to prune it, searching that graph can discover solutions. An example of this type of approach can be found in the rule and production based systems such as LOOS (Flemming 1994:5). It contains a generator able to accept a layout and find all possible ways of adding a new object. A tester evaluates a layout generated in this way and a controller mediates between these two components. After each generate-and-test-cycle, the designer selects the next layout to be expanded based on the evaluations produced by the tester.

- *Hierarchical refinement*: Refine skeletal designs from the top down. Hierarchical refinement assumes that there are really only a few basic types of designs. If the problem can be classified, a design template can be instantiated, and propagating constraints can solve variables. An example of this is the simple cut-and-paste approach developed by the Division of Building Technology of the CSIR in South Africa for health facility design. It is based on a large set of templates at various levels and assists unskilled designers to rapidly design a hospital or a clinic.

- *Transformational approach*: In this approach design is claimed to be a mapping from function to structure. Just as Fourier and Laplace transforms map from one domain that is difficult to reason in to another that is more tractable, the transformational approach to design suggests decomposing functions and mapping primitive functions onto structures. Conradie and Küsel (1999) experimented with this in the precedent system AEDES discussed in Chapter 4.

- *Case-Based Design*: The case-based and analogical approaches assume that the problem being solved is probably similar to one that was seen before. If a historical case can be retrieved, a solution can be found by transferring directly from that previous case. An example is the ARCHIE-2 system (Goel *et al*. 1991; Kolodner 1993) that uses similar solutions from the past to solve the current problem.

Currently the most promising AI solution is the use of design cases. This is empirically validated successful solutions and failures to design problems from the past. If structured design methodologies are to be used then design knowledge generated should be stored in such as way as to expedite future designs. This is also one of the key objectives of Knowledge-Intensive CAD where attempts are being made to elevate CAD systems beyond only electronic drawing boards. Mäntylä (1995: 3) states that a key objective for the logistical management of information is reuse of existing information. Ideally all (design) information created or learned should be made available to later use in a correct, useful and timely fashion.

### 3.2.2 Artificial intelligence and design

In the late 1950s Allen Newell and Herbert Simon proved that computers could do more than calculate. Marvin Minsky, head of the Massachusetts Institute of Technology (MIT) Artificial Intelligence (AI) project at the time, announced with confidence that within a generation the problem of creating Artificial Intelligence would be substantially solved. Then suddenly the field of AI ran into unexpected difficulties. The trouble started with a failure of attempts to program an understanding of children's stories. The program lacked the common understanding sense of a four year old and no one knew how to give the program the background knowledge necessary for understanding even the simplest stories. An old rationalist dream was at the heart of the problem. AI is based on the Cartesian idea that all understanding consists in forming and using appropriate symbolic representations. For Descartes, these representations were complex descriptions built up out of primitive ideas or elements.

Dreyfus (1993:*xi*) states *"Common-sense understanding had to be represented as a huge data structure comprised of facts plus rules for relating and applying those facts."*

AI struggles with essentially three central problems (Dreyfus 1993:*xviii*).

- How everyday knowledge must be organised so that inferences can be made.
- How skills or know-how can be represented as knowing-that.
- How relevant knowledge can be brought to bear in particular situations.

Dreyfus (1993:*xxviii*) states that *"Heidegger, Merleau-Ponty, and the gestaltists would say that objects appear to an involved participant not in isolation and with context-free properties but as things that solicit responses by their significance."*

*"What we really need is a system that learns on its own how to cope with the environment and modifies its own responses as the environment changes. To satisfy this need, recent research has turned to an approach sometimes called 'reinforcement learning'."* (Dreyfus 1993:*xxxix*)

*"The point is that a manager's expertise, and expertise in general, consists in being able to respond to the relevant facts. A computer can help by supplying more facts that the manager could possibly remember, but only experience enables the manager to see the current state of affairs as a specific situation and so see what is relevant. That expert know-how cannot be put into the computer by adding more facts, since the issue is which is the current correct perspective from which to determine which facts are relevant."* (Dreyfus 1993:*xlii*)

Feigenbaum makes the following comments in his analysis of MYCIN, a program developed by Shortliffe in 1976 for diagnosing blood and meningitis infections and recommending drug treatment (Dreyfus 1993:28).

*" He conscientiously notes that the experts themselves are not aware of using rules:*

*...Experience has also taught us that much of this knowledge is private to the expert, not because he is unwilling to share publicly how he performs, but because he is unable. He knows more than he is aware of knowing. (Why else is the Ph.D. or the Internship a guild-like apprenticeship to a presumed 'master of the craft'? What the masters really know is not written in the textbooks of the masters.) "*

The author tested the translation capabilities of a translation program freely available on the Internet. The result is really amazing, however upon closer inspection certain inherent and fundamental problems become apparent.

The following slightly technical paragraph from a recent German conversation class was submitted to the translator.

*" Die Stadt Frankfurt am Main*

*In der Stadt Frankfurt gibt es heute viele grosse Bürogebäude und in der Umgebung findet man viel Industrie. Die Farbwerke Höchst, wo Farben, Lacke und andere Chemikalien erzeugt werden, sind in der Nähe von Frankfurt zu Hause. Andere grosse industrielle Konzerne sind Siemens und Halske AG., Hartmann und Braun (Elektroinstrumente) und Mouson (Kosmetik). Auch die Glas- und Porzellanindustrie ist bedeutend.*

*Frankfurt ist auch ein kultureller Mittelpunkt und hat natürlich eine Universität und verschiedene Hochschulen. Das Städel ist die städtische Kunstgalerie und besitzt viele Kunstschätze. "*

After a few seconds the translator responded back with the following partially correct answer.

*" The city Frankfurt/Main*

*In the city Frankfurt gives it today many large office buildings and in the environment finds one much industry. The inking attachments Hoehst, where colours, lacquers and other chemicals are produced, are in the proximity from Frankfurt at home. Other large industrielle of companies are Siemens and Halske AG, hard man and brown (electrical instruments) and Mouson (Kosmetik). Also the glass and porzellanindustrie are important.*

*Frankfurt is also a cultural focal point and has naturally a university and different universities. The Staedel is the urban art gallery and possesses many art treasures. "*

Upon analysis of the results it is apparent that the problems mentioned by Dreyfus are real. The wrong translation of the phrase *"gibt es heute"* indicates that the context or idiomatic expression was not understood. The most serious error was the company *"Hartmann und Braun"* that was translated as *"hard man and brown"*. This indicates that purely mechanistic parsing was used in this case without any higher level contextual comprehension. It is also apparent in the short paragraph that the translator had difficulty with certain technical terms such as *"porzellanindustrie"*. It is the author's experience, and Dreyfus confirms it, that AI is only successful in small well-defined domains. The above translation does not make sense on its own. If AI is to be successful at all it is mandatory that context be well understood. Nobody has yet succeeded in devising an algorithm that can accurately summarise written text or a book, because this process would require exceptional contextual understanding. This is indeed one of the most difficult problems known.

Dreyfus (1993:*xxx*) notes *"It seems highly likely that the rationalist dream of representationalist AI will be over by the end of the century"*.

Designers and CAD researchers became interested in AI for two reasons. The first is the influence of structured programming as propagated by Dahl, Dijkstra and Hoare (Flemming 1994:1). Computer programming is seen as a process of step-wise refinements where program specifications are developed through several levels of abstraction. The specification is complete at any level. Transitions from one level to the next consist of expanding the program by adding greater detail. The prescriptions of structured programming are impossible to follow in many design situations because they presuppose that the task at hand is well understood and amenable to algorithmic treatment.

The second reason was due to the frustrations with the unintelligent nature of commercial CAD systems. Even today CAD is contributing very little to the initial and most demanding stages of design. In an attempt to solve the latter AI was applied. AI is generally concerned with tasks whose execution appears to involve some intelligence if done by humans. Design falls into this category.

AI research can be divided into two broad approaches.

- *Understanding of the human brain.* Computer models in this tradition represent a model or simulate human cognition and succeed to the degree to which they emulate human performance.
- *Intelligent systems.* Systems that perform intelligent tasks effectively without concerns for how faithfully the model simulates human performance or cognition.

The efforts in the first category are theoretically motivated and must seek empirical acceptance. Efforts in the second category are practically motivated and must stand the test of practical usefulness. Computers that work exactly like people are unlikely to do better than people. CAD tools, whether AI based or not, should always be seen as a complement to human designers assisting them in tasks where they perform less well, but do not compete in areas that the human brain performs well. Programs that assist in design are most useful in the following areas:

- Suggest possibilities to designers they have not thought of.
- Remind them of things they might have forgotten.

The author will attempt to prove that in addition to the two possibilities a third option exists. This is where intelligent components are used to facilitate the manipulation of complex design information in a convenient environment to facilitate concept selection and design experimentation during the early phases of design. During this phase the designer is often confronted with incomplete information and designs could very easily change. At the same time decisions taken during this phase will significantly influence operational characteristics.

Flemming (1994:21) states that the fixation of some AI researchers on processes rather than results is puzzling. Chess-playing programs were initially considered a legitimate AI topic. Recently the world chess champion, Kasparov was beaten by the super-computer Deep Blue. This was not achieved by imitating chess players, but rather by a more efficient generate-and-test approach that results, in part, simply from hardware improvements. In addition there was behind-the-scenes expert intervention. Bellman (1978:144) came to the conclusion that the human brain remains far above anything that can be mechanised. Oksala (1994:27) states that many architectural problems are life-long and in a theoretical sense typically non-terminating. Architectural products are often so complex with respect to environmental situations that we can only describe them partially.

#### 3.2.2.1 Life cycle enabled design ontology

In the design and implementation of software intended to support the design and construction environment, ontology plays an important role. Simoff *et al.* (1998:23) mention that ontology originated in philosophy as a systematic account on the nature and the organisation of reality. Currently ontology is considered to be a branch of metaphysics addressing issues such as the categorical structure of reality. There is no ontology that is accepted as the definitive categorical scheme. A typical categorical scheme has a hierarchical structure with the most general entity at the top of the hierarchy.

Simoff *et al.* (1998:23) mention that the concept of ontology entered the field of artificial intelligence as a formal system for representing domain concepts and their related linguistic realisations by means of basic elements.

Unfortunately there is growing confusion about the meaning of the term in the context of its usage in AI in design. Presently the term is so wide that it ranges from the ISO STEP object model description (a hierarchical interoperability standard) to concept structures for sharing ideas. The application of ontology for the description of design domain faces additional difficulties due to the interdisciplinary and evolutionary nature of the domain (Simoff *et al.* 1998:24).

In the precedent systems PREMIS and AEDES three fundamentally distinct ontologies can be identified.

- *Location (property)*. This is a hierarchical structure that expresses the hierarchical relationship of locational entities. It starts with the *country* definition and goes right down to *shared space* (Figure 11).
- *Administration*. This is related to the activities and administrative uses that are made of these locational entities for managerial, classification or maintenance purposes.

- *Graphical objects*. The graphical database contains a set of structured entities that links to the alphanumeric relational database.

Due to the fact that *location/ property* is pivotal to Facilities Management portion of the life cycle this world was viewed as primary in PREMIS. *Administration* is broken down into the main categories *organisational structure*, *legal*, *people* and *construction elements* (Figure 12). The relationship between *graphical objects* and the *location alphanumeric* category is maintained by means of a system of implicit linking (Appendix A). In this system a graphical object has a relationship with an alphanumeric record by virtue of the fact that the graphical object name is the same as the concatenation of the key fields in the relevant Relational Database Record.

In an attempt to structure briefing and design Conradie and Küsel (1999) used an oversimplified ontology in the AEDES prototype of *active* and *passive* requirements and functions. Passive requirements are related to physical elements such as *structure*, *services*, *finishes*, *fittings*, *furniture* and *equipment*. Active requirements are activities that were given activity function names such as *enable ablutions*. In this approach SE principles as well as functional decomposition were explored.

Simoff *et al.* (1998:28) suggest an ontology that delineates the categories of building design space as *activity* and *space*. An *activity* consists of *equipment*, *service*, *time*, *performer*, *consumer* and *constraints*. Space consists of *geometry*, *divider*, *link* and *constraints*. In this model relations define the explicit connection between entities.

In Product Data Modelling (PDM) that is essentially the briefing and design phase of the product life cycle another important world can be identified:

- *BPM structure*. This is a traceable hierarchical framework for organising design knowledge and documentation on every stage of the design process. A good example of this is the Industry Foundation Classes (IFC™ Release 2.0) as defined by the International Alliance of Interoperability (IAI). This is essentially a hierarchical object structure that describes structural interactions, but also attempts to facilitate interoperability between different systems in the construction industry. This model is not as comprehensive as the ISO STEP object standard (Figure 13).

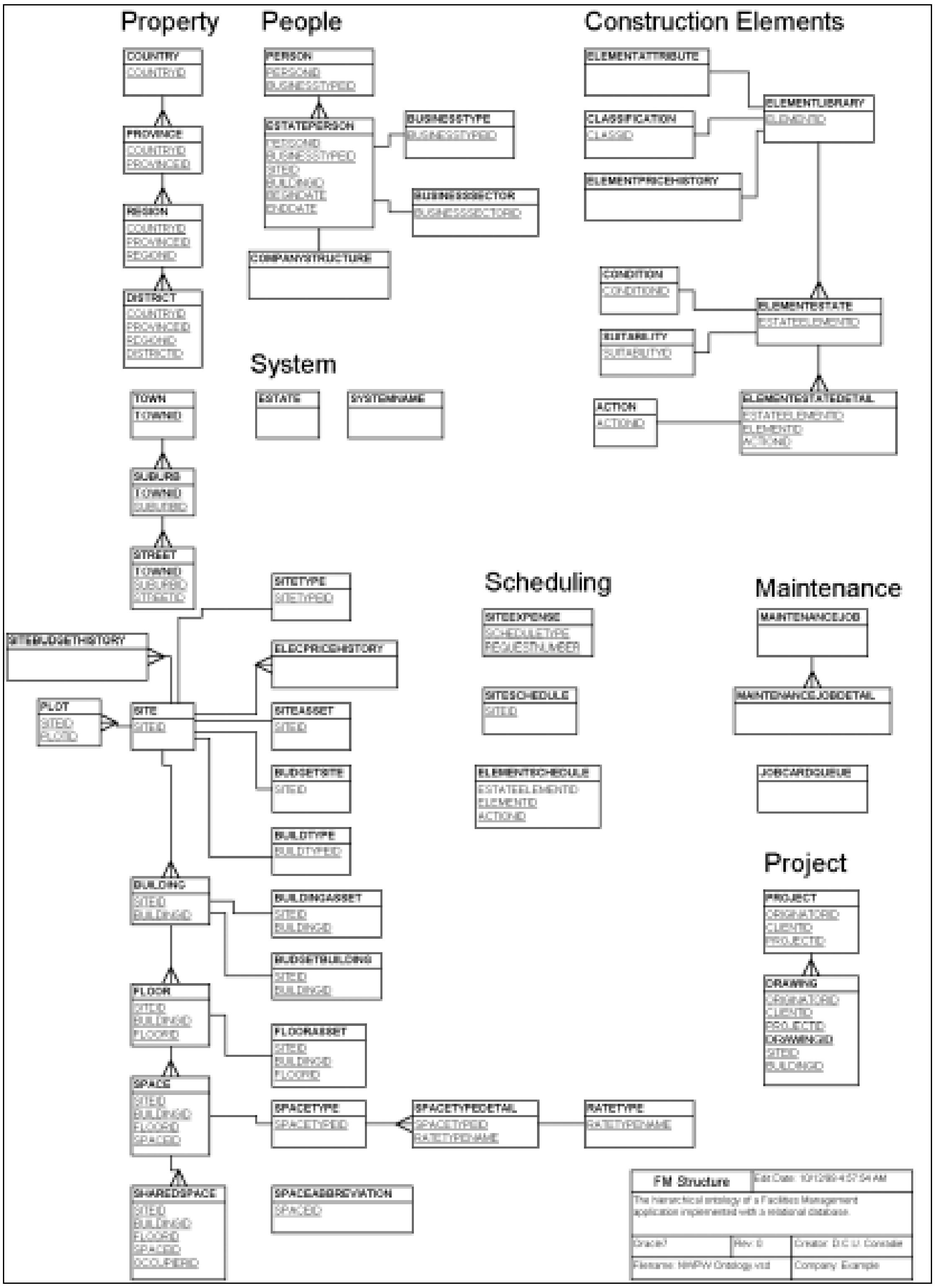


Figure 11: Typical hierarchical relational database structures used in a Facilities Management system (Author)

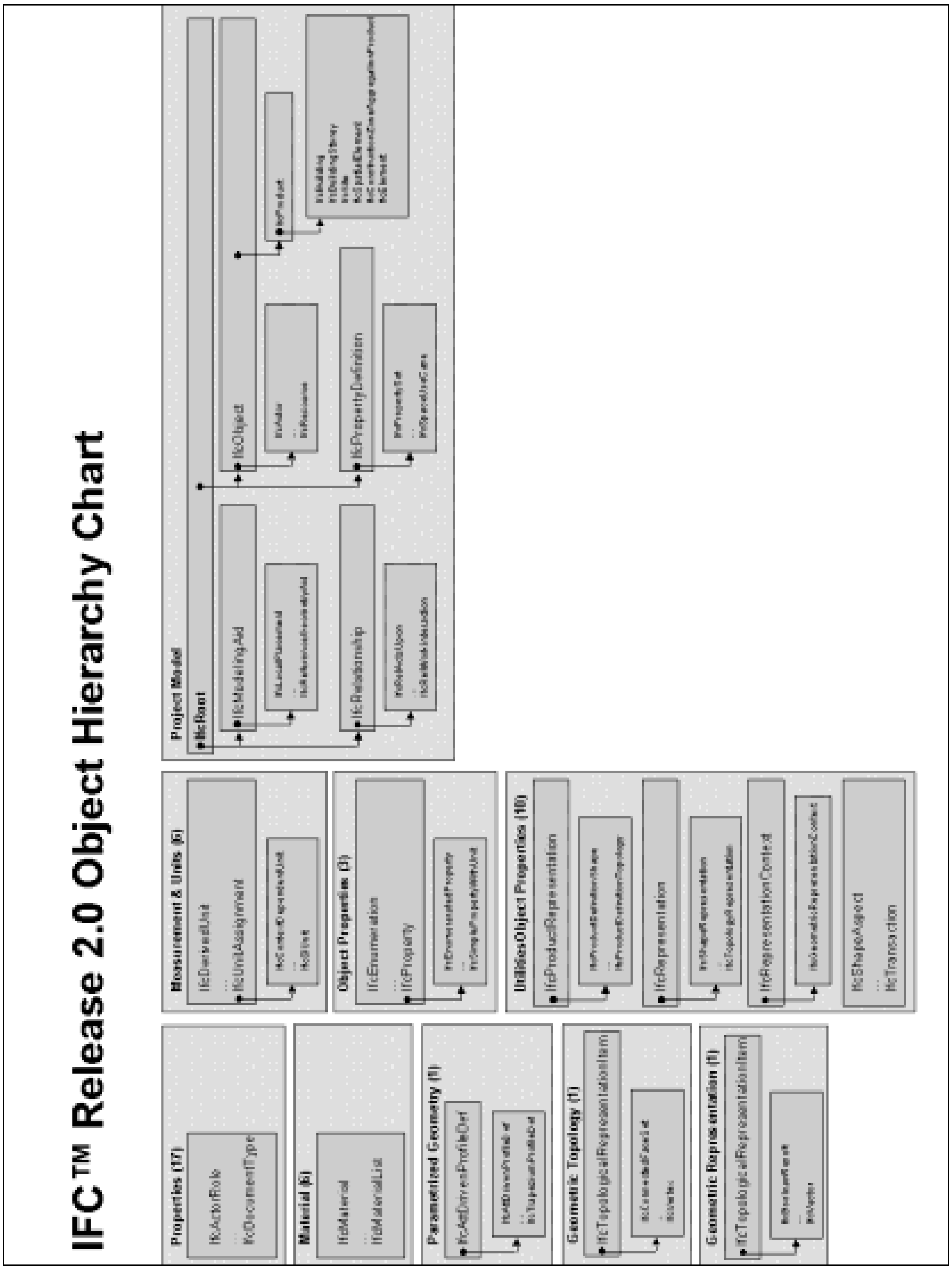


Figure 12: IAI, Industry Foundation Classes Release 2.0 Object Hierarchy (Author)

## 3.2.3 Problem-solving architectures

Today the following general design problem-solving strategies exist (Flemming 1994):

- Top-down strategies that develop a design specification through several levels of abstraction.

- Bottom-up strategies that construct a design incrementally in small steps more or less at the same, final level of abstraction.
- Middle-out strategies that start with a highly structured description and, transform it to satisfy given requirements.

Within each of these strategies a function or behaviour driven approach could be distinguished from a form driven approach. These distinctions correspond approximately to the goal- and data-driven approaches in AI.

#### 3.2.3.1 Top-down strategies

**1. Function-driven strategies**

The logic design component selects functional components from a knowledge base that contains descriptions of individual components and templates that tell the system how to design with them. This synthesis proceeds through the levels of a functional hierarchy. At the highest level, it derives the overall system architecture in terms of functional subsystems and ensures correct interconnections between subsystems. The next level inherits the characteristics of the components determined at higher levels and can split single functional components into successor parts. When the lowest level is reached the individual subsystem components and their connections are known. The AEDES prototype system is an example of this type of functionally based system.

**2. Form-driven strategies**

This strategy is closely related to the type of structures that the IAI (Figure 12) proposes. It is a decomposition of the building structure into the various sub-structures and materials. The fundamental approach is to build a model of a building and to generate all documentation from the 3D model. The process to design a structure for a specific building starts from specifications of the overall building form and type. It successively selects component types and materials at each level in the hierarchy. The choices available at each step can be found by one of three methods:

- Selection from a pre-defined, finite set of alternatives.
- Synthesis by further decomposition.
- Computation based on rules or numerical calculations.

Constraints can be defined to avoid certain combinations of decisions. A design approach in which the parts for assembly are selected from a predefined set is also called *configuration design*.

#### 3.2.3.2 Bottom-up strategies

**1. Function-driven strategies.**

This approach takes individual functional, behaviour or performance specifications and derives a description of a design incrementally by taking these specifications into account. Flemming (1994) states that this approach is rare because the interactions between performance indicators and design variables are so complex that it does not generally render this approach feasible. One particular system (WRIGHT) avoided these problems by using a system of disjunctive constraints that translate the desired behaviour characteristics into constraints in the design variables. This system determines all feasible ways of satisfying the constraints incrementally using constraint satisfaction techniques developed in AI.

**2. Form-driven strategies**

Form-driven, bottom-up strategies are employed by the classical incremental generate-and-test approaches that generate a design in small steps. The intermediate evaluations are used to direct the process into the most appropriate direction. Typically it would contain a generator that is able to accept a layout and find all possible ways of adding a new object. A tester evaluates a layout generated in this way and a controller mediates between these two components. After each generate-and-test-cycle the controller selects the next layout to be expanded, based on the evaluations produced by the tester. An early experimental system that used this approach is LOOS. It is unlikely that future Case-Based Design methodologies would use this type of approach, because it is so tedious. The computer attempts to execute tasks that a human designer can do just as well.

#### 3.2.3.3 Middle-out strategies

**1. Function-driven strategies.**

In this approach a system starts with a highly structured description of the desired behaviour of a design. This description is transformed, at the same level of abstraction or granularity, into a physical description. An example of this is a system that accepts a graph-based description of an algorithm to be executed by a computer chip and transforms this description into a collection of hardware components and their connections.

**2. Form-driven strategies**

This strategy starts with a highly detailed and structured design description and refines or adapts it to the given context. Examples of this are the ARCHIE and ARCHIE-II systems described in detail by Kolodner (1993). ARCHIE is an interactive prototype *Case-Based Reasoning* (CBR) system for the design of buildings such as libraries and courthouses. It supports the construction and evaluation of solutions. Users specify their problem description and/ or solution description. The system retrieves and displays past designs and provides suggestions and warnings. In support of evaluation, the system computes potential outcomes and retrieves and displays past designs with similar outcomes. ARCHIE showed that design cases could be very large and need to be decomposed into smaller units. Libraries of design cases can be useful but may need to be supplemented with other types of design knowledge. Practical support systems need usable interfaces to allow easy access to relevant information. The most important lesson learnt is that the operation should be kept simple. ARCHIE is a useful precedent for the present study even though Kolodner (1993:162) described ARCHIE as a failure.

Architectural design systems based on CBR must solve two major system design and implementation problems:

- *Indexing*. An indexing system must be designed to facilitate retrieval of stored cases so that the most appropriate ones can be retrieved in the new design situation.
- *Adaptation*. They must support the refinement or adaptation of an existing case to the new situation.

## 3.2.4 Case-based design

### 3.2.4.1 Introduction

A solution stored for possible reuse at a later time is called a *case* in the AI literature. The ability to "frame" a problem is what differentiates great from ordinary designers. It is the ability to distinguish between the vital few and mundane many design factors that leads to a good design. Kolodner (1993:13) defines a *case* as a contextualized piece of knowledge representing an experience that teaches a lesson fundamental to achieving the goals of the reasoner. Rather than viewing reasoning primarily as a composition process, *Case-Based Reasoning* (CBR) views reasoning as a process of remembering one or a small set of concrete instances or cases and basing decisions on comparisons between the new situation and the old instance. This view has important implications:

- CBR emphasises the use of concrete instances over abstract operators. It regards large chunks of composed knowledge as the starting point for reasoning. Though there may be smaller and more abstract chunks of knowledge in memory, they derive from cases and are thus secondary to them (Kolodner 1996:361).

- CBR emphasises manipulation of cases over composition, decomposition and recomposition processes. Reasoning by use of cases comes first and composition of operators is of secondary importance (Kolodner 1996:364).

Of all the AI methods available today, Case-Based Design (CBD) is the most promising with regards the storage of previously synthesised design solutions. CBD is a sub-set of CBR that is aimed specifically at design using CBR methods.

CBD facilitates the provision of a comprehensive design database of past solutions that designers will not remember on their own. CBD has distinct advantages over other AI techniques such as *Knowledge-Based Systems* and *Models.*

The CBR paradigm has a bias against problem decomposition and recomposition implied by composition of operators, because composition is a highly complex process. When problems are entirely decomposable into noninteracting parts, decomposition and recomposition are easy. As problems become less and less decomposable into non-interacting parts, recomposition becomes harder and harder. Traditional methods must be stretched beyond their original intent to deal with these problems. Such problems, which are called *barely decomposable*, can be more efficiently solved by methods that do not have to decompose them (Kolodner 1993:16).

Kolodner (1993) is of the opinion that engineering and architectural design is almost entirely a process of adapting old solutions to fit a new situation or merging several old solutions to do the same. Carrara *et al.* (1994) agree with this viewpoint when they characterise design as:

1. Defining a set of functional objectives that ought to be achieved by the design artefact.
2. Constructing design ‘solutions’ which, in the opinion of the designer, are (or should) be capable of achieving the predetermined objectives.
3. Verifying that these solutions are internally consistent and that they achieve the objectives.

Richens (1994:309) strongly disagrees with this point of view when he claims that architectural objectives usually include functional ones, but are dominated by less definable intentions. Flemming (1994:22) states that attempts to introduce machine innovation and creativity are red herrings. Oksala (1994:41) expresses the opinion that it is realistic to design machines that work as architectural design assistants and are capable of redesigning work according to given rules.

It is essential that this database be built up during the normal activities of a design firm. If a designer has generated a solution he should be able to store it literally with the push of a button. Most experimental prototype systems at the moment rely on independent and separate processes that may require the assistance of an expert that is intimately familiar with the technicalities of indexing and retrieval. The author is of the opinion that *Case-Based Systems* would be practical when designers themselves are actively involved in the modification of a case and its storage for re-use. This machine/ designer relationship uses the best of both worlds.

The author argues that CBD is a valid option for the following reasons:

1. Experts (domain experts) using any preferred front-end design method build up the corpus of knowledge. It does not preclude traditional methods. Structured methods would be more efficient.
2. Unlike other structured methods CBR allows a problem to be solved as a complete unit. This is closer to the holistic synthesis of design problems that is dominant in architectural design.
3. Successful precedents in an architectural environment already exist (Kolodner 1993).

If a similar problem has been solved previously, it can provide the glue that holds barely decomposable problems together. Rather than dealing with hard recomposition problems, the reasoner only has to address those parts of the old solution that do not fit the new situation.

Case-Based Reasoning (CBR) is an approach to knowledge, memory structure and reminding that is based upon modelling experiential knowledge. It is characterised in the literature as a problem solving approach of a reasoner, which makes inferences from previous solutions which are adapted to current situations. It has demonstrated its usefulness in domains where experience is strong, but the domain model is weak or poorly formalised (Oxman *et al.* 1994). Rather than duplicating human cognition, these models attempt to capture the essence of the human cognitive processes and to explicate the principles of their operation. A new generation of Knowledge-based System can potentially work in a partnership relationship with the human designer. The objectives of support or aid systems have been defined as to enhance human decision making by suggesting alternatives, predicting consequences and conveniently grouping together the information that goes into decision making.

There are many different types of solutions. The solution to a design problem is the artefact that was designed. With a solution in place, a reasoner that retrieves a case can use its solution to derive a new solution. Solutions also have other components that aid adaptation. The following list from the research community as interpreted by Kolodner (1993:154) is useful:

- The solution itself.
- The set of reasoning steps used to solve the problem. This was well addressed in the AEDES prototype.
- The set of justifications for decisions that were made in solving the problem.
- Acceptable solutions that were not chosen and the reasoning and justifications that go with them.
- Unacceptable solutions that were ruled out and the reasoning and justification that go with them.
- Expectations of the result of deployment of the solution.
- Things that went wrong with the previous solution.

### 3.2.4.2 Advantages of a Case-Based Reasoner?

CBR has several advantages that give an indication when it should be used. The list below has been collated and adapted from Kolodner (1993). The following advantages can be identified:

1. CBR reasoning allows the reasoner to propose solutions to problems quickly, because it avoids the time necessary to derive those answers from scratch.
2. CBR allows a reasoner to propose solutions in domains that are not completely understood. This is of particular importance to the advanced planning that is necessary to design and build complex facilities such as hospitals.
3. Remembering previous experiences is particularly useful in warning of the potential for problems that have occurred in the past, alerting a reasoner to take action to avoid previous mistakes.
4. CBR can be used as a communication tool between designers and other less design literate participants.
5. Cases help a reasoner to focus his reasoning on important parts of a problem by pointing out what features of a problem are the important ones.
6. A CBR system can be made to learn. In CBR problem solving efforts are saved to expedite future work. Learning is a natural consequence of problem solving efforts. CBR systems can be designed in such a way that they adapt to changes in their environments by means of adaptive fuzzy sets, discussed below. The system can continue to collect cases after deployment.
7. When CBR is used to solve problems, solutions can be justified by the cases they are derived from. In a domain where it is difficult to evaluate solutions objectively such as architectural design, CBR has the advantage of providing illustrations of the effects of particular solutions.
8. CBR can be designed to anticipate potential problems as natural part of their reasoning. Unsuccessful experiences with past solutions can be used in case-based systems to anticipate possible problems that might result from solving a problem a certain way. In general this capability adds efficiency. In architectural design anticipation of problems is critical.
9. CBR provides a way for designers and computers to interact in a realistic way. CBR is fundamentally inspired by human behaviour. Certain tasks in design such as the calculation of energy consumption or acoustic performance is easier for a computer to achieve, whereas aesthetic design decisions is best decided by the designer. Designers are good with creative reasoning, but poor at remembering the full range of applicable cases. Humans tend to be biased in their remembering or as novices they not yet had the experiences they need to solve the problem. During an interview of the professional team involved in a large and complex construction project this fact was emphasised.
10. The knowledge acquisition for a CBR system is natural. Concrete examples rather than piecemeal rules can be used. Experts (experienced practitioners) find it difficult to report the knowledge they use to solve problems. They are quite at home reporting their experiences and discussing the ways in which cases are different from one another.
11. CBR should be considered when it is difficult to formulate domain rules but cases are available. Formulating rules is difficult in weak-theory domains such as architectural briefing and design. In this domain knowledge is incomplete, uncertain or inconsistent. It is impossible to formulate rules when there is a great amount of variability in design situations that have the same outcome.
12. CBR can be considered when rules that can be formulated require more input information that is normally available. This may be due to incomplete specified problems or the fact that the knowledge required is not available at problem-solving time. This is often the case in the construction industry and fast track projects where all project information is not available up-front.

13. CBR should be considered when it is expensive to use rules because the average rule chain is long.
14. CBR should be used when generally applicable knowledge is not sufficient to solve a problem. This could be due to the fact that knowledge changes with context or because some of the knowledge required solving the problem is used only under special circumstances.
15. CBR should be considered when a case library already exists. In the present study some hospital design cases (starter kits) are already available, albeit in an unstructured format.
16. When no fast computation method exists for deriving a solution from scratch, CBR allows new solutions to be derived from old ones. In the case of the basic hospital starter kit set developed at the Division of Building Technology at the CSIR, simple hospitals but different hospital designs can quickly be built by means of different exemplar department and architectural units.
17. When there is no fast computational method for evaluating a solution or when there are so many unknowns that evaluation methods are unusable or difficult to use, CBR provides an alternative.
18. CBR allows evaluation of solutions when no algorithmic method is available for evaluation.
19. Cases are useful in interpreting open-ended and ill-defined concepts.

### 3.2.4.3 The disadvantages and caveats of Case-Based Reasoning

CBR has several disadvantages and caveats in architectural design that should also be considered. The list below has been collated and adapted from Kolodner (1993):

1. CBR requires cases. Traditionally the effort in building a CBR system went into case collection. It is apparent from a study and interviews[1] with the designers of ARCHIE that it was an enormous effort. To be successful in the architectural profession and the construction industry it should not require such extraordinary efforts. The case library should be automatically assembled during the normal professional design activities.
2. For CBR to be useful and reliable, cases with similar problem statements should have similar solutions. CBR is based on the premise that situations recur in a predictable way. Adaptation modifies old solutions to fit new situations. If a domain is discontinuous where similar situations require wildly different kinds of solutions, then CBR cannot be used and would be misleading. This is unfortunately only partially true in architecture. Creative designers do not always solve related design problems in a similar way.
3. CBR solutions are not guaranteed to be optimal. The full range of possible design solutions is usually not explored in a CBR system intended for design support. Optimal or more creative solutions may be missed. This is a problem in any heuristic system such as TRIZ that is also discussed in the present study. The designer cannot escape his responsibilities, however the CBR system will remind him of design aspects he might have forgotten.
4. An inexperienced case-based reasoner might be tempted to use old cases blindly, relying on previous experience without validating it in the new situation.
5. A case-based reasoner might allow cases to bias him or her too much in solving a new problem.
6. Case libraries require considerable storage space. In the design of CBR systems special consideration must be given to ensure a long life of the case with changing technology. A large sum of money in terms of intellectual capital, time and effort is encapsulated in the case library. Persistence of data is therefore of paramount importance.
7. Inexperienced people are often not reminded of the most appropriate sets of cases when they are reasoning.

[1] Janet Kolodner and Craig Zimring personal communication during April 2000.

### 3.2.4.4 Case-based Reasoning compared with other methods

The CBR/ CBD cycle (Kolodner 1993:18) has striking similarities with the product development method of concept selection proposed by (Pugh, 1996; Ulrich *et al.*, 1995) (Figure 13). In generalised terms the CBD cycle is the case equivalent of concept selection.

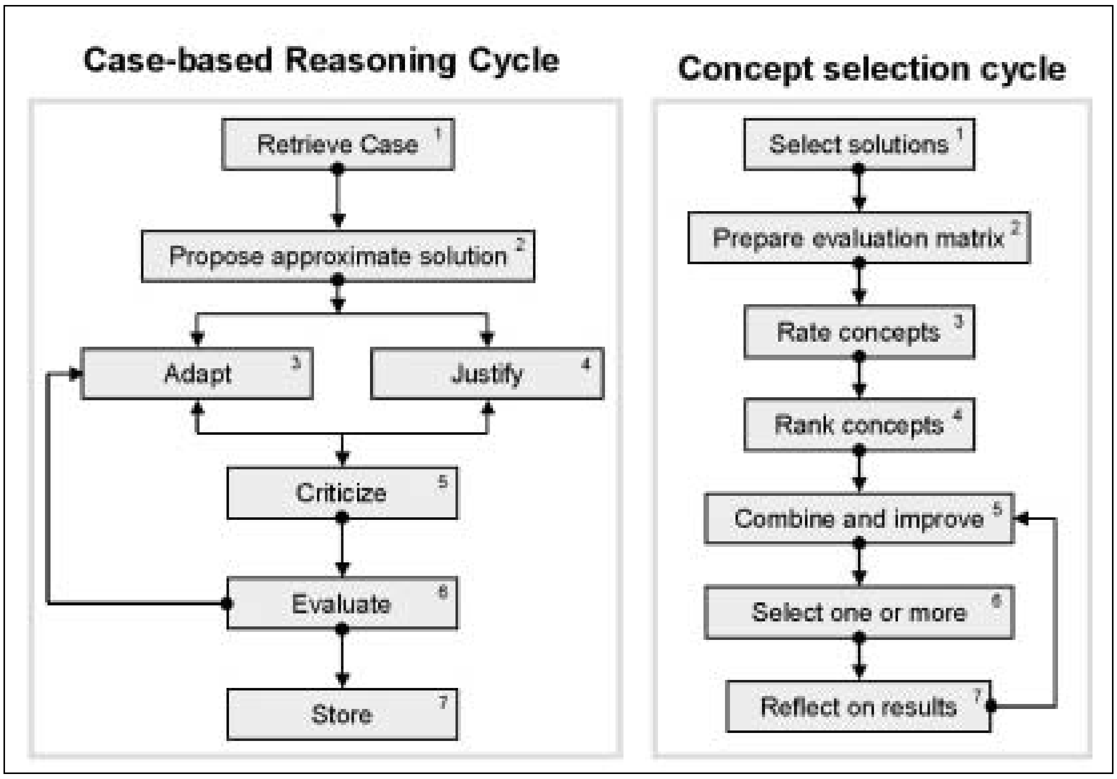


Figure 13: Case-Based Reasoning compared to concept selection (Collated by author from Kolodner (1993:18), Ulrich *et al.* (1995) and Pugh (1996) )

The typical stages of the CBD cycle are (Kolodner *et al.* 1996:35):

1. *Retrieval*. Partially matching cases must be retrieved to facilitate reasoning. This is called *case retrieval.* The case was created in the first instance by a *case storage* process also called *memory update*.
2. *Solution proposal*. In problem-solving CBR, a ballpark solution to the new problem is proposed by extracting the solution from the retrieved case.
3. *Adaptation*. This is the process of altering an old solution to fit it to the context of the new situation.
4. *Criticism*. This is a critical analysis of the new solution before applying it.
5. *Justification*. This is the process of creating an argument for the proposed solution, done by a process of comparing and contrasting the new situation with prior cases. Sometimes justification might by followed by a criticism step in which hypothetical situations are generated and the proposed solution applied to them in order to test the solution.
6. *Store (memory update)*. The new case is permanently saved for future use.

The following table compares *Case-Based Reasoning*, (CBR), *Rule-Based Reasoning* (RBR) and *Model-based Reasoning* (MBR)[1].

[1] Janet Kolodner is of the opinion that CBR, MBR and RBR form a continuum. Personal communication 14 April 2000.

Table 4: A comparison between Case-Based, Rule-Based and Model-based Reasoning (Collated by author)

| **Case-Based Reasoning** | **Rule-based Reasoning** | **Model-Based Reasoning** |
|---|---|---|
| Cases in case libraries are constants that describe the way things work. | Rules in rule bases are patterns. | Store causal models of devices or domains. |
| Cases are retrieved that match the input partially. | Rules are retrieved that match the input exactly. | |
| Cases are retrieved first, approximating the entire solution at once, then adapted and refined to a final answer. | Rules are applied in an iterative cycle of microevents. | |
| Cases are large chunks of domain knowledge, quite likely redundant, in part, with other cases. Based on idiosyncratic knowledge, specific to episodes but mostly not normative. Provides methods for constructing solutions. | Rules are small, ideally independent but consistent pieces of domain knowledge. | Emphasise general knowledge that covers a domain. Models hold knowledge needed for validation or evaluation of solutions but do not provide methods for constructing solutions. |
| CBR can be used both when a domain is well and not so well understood. In the latter case it assumes the role of a generalised model. | Not applicable | Is used when a domain is well enough understood to enumerate a causal model. |
| Provides for efficient solution generation and evaluation is based on the best cases available. | Not applicable | Provides a means of verifying solutions, but solution generation is unguided. |
| Needs a means of evaluating its solutions, guiding its adaptation and knowing when two cases are similar. | Not applicable | Models provide a means of evaluating its solutions. |

These differences led to differences in knowledge acquisition. In RBR, knowledge is extracted from experts and encoded in rules. This is often difficult to achieve. In CBR most (but not all) knowledge is in the form of cases. CBR needs adaptation rules and similarity metrics and more types of knowledge, but knowledge is easier to acquire.

Both MBR and CBR were developed as methods for avoiding reasoning from scratch. Both compose knowledge into large chunks and reason using large chunks. The differences have mostly to do with the content of the knowledge used and the conditions of applicability for each.

### 3.2.4.5 Types of Case-Based Reasoners

Kolodner (1993) distinguishes between automated reasoners and retrieval-only aiding and advisory systems. Numerous cases can be found in the literature to illustrate the former type that can achieve numerous diverse tasks. Typical examples are:

1. CHEF is a case-based planner. Its domain is recipe creation. Recipes are viewed as plans.
2. CASEY is a case-based diagnostician. It takes as its input a description of its new patient, including normal signs and presenting signs and symptoms. Its output is a causal explanation of the patient’s disorders.
3. JULIA is a case-based designer that works in the domain of meal planning.

4. HYPO is an interpretative reasoner that works in the domain of law. It takes as input a legal situation and as its output it creates an argument for its legal client.
5. PROTOS implements both case-based classification and case-based knowledge acquisition. Given a description of a situation or object, it classifies the situation or object by type.
6. CLAVIER is a manufacturing industry related system for configuring the layout of composite aeroplane parts for curing in an autoclave. It is used at Lockheed in California.
7. ROBBIE (Re-Organisation of Behaviour by Introspective Evaluation) combines a case-based planner with an introspective component. It was used to simulate an intelligent agent travelling in a limited world of a number of street blocks. The agent had to conform to certain basic rules. The agent could intelligently work out alternative rules if an unforeseen obstacle came in the way (Fox 1995).

The former group is where an application in an architectural domain is most likely to achieve success. CBR fits well with the way that designers work. People use CBR naturally in much of their everyday reasoning. Kolodner (1993) provides existing examples that are precedents for the present study. These examples are all retrieval-only aiding and advisory systems. The first one is a hypothetical architect’s assistant. ARCHIE and ARCHIE-II are useful precedents of prototype systems that give direction to the present study. ARCHIE-II uses the concept of design stories. Some stories in ARCHIE-II tell about design features that did not work and what could be done to remedy the situation. Others report on features that were successful. Users first describe to the system the problem they are working on. The system subsequently retrieves buildings that are similar to the desired new one. The user can then display the building or part of a building that is retrieved. He is shown a floor plan surrounded by annotations. These annotations describe the various design features.

Domeshek *et al*. (1994) describe the MIDAS (Memory for Initial Design of Aircraft Subsystems) system. This system used ARCHIE as a precedent to support early design of aircraft subsystems. Both ARCHIE-II and MIDAS use the Design-MUSE shell that eases construction of case-based design aids. An important goal of this system was that domain experts should be able to maintain it, rather than AI experts.

Oxman (1994) recognises four cognitive approaches for modelling design case knowledge:

- Generic models (model-based)
- Associative models
- Exemplar models
- Precedent

**3.2.4.6 Generic models**

A design space is essentially a delineation of a class of things. That is designs conforming to particular meanings and a particular syntax. Knowledge is used to define classes of designs called generic designs. It is often convenient to make the generic nature of knowledge explicit. Rather than using grammatical rules, a design space may be defined in terms of a class description called a *generic model*. This is done by listing all properties of the class, including the ranges of properties that an instance may take and also the interrelationships between properties. A small house can be defined in terms of its generic form and attributes. The graphic structure contains implicit information about the essential properties of the class. The properties could also be listed. An example is the list of allowable rooms it may have. It may also be stated that it includes items such as a roof and a front door.

In some cases a certain design instance is said to typify a class. It embodies the features of its class in which we are interested. Such a design is said to be an *archetype*. The concept of an *archetype* is useful because we prefer to think in terms of instances rather than in terms of the abstract world of classes of things.

In design the term *prototype* is also used. This is generally seen as a design from which other designs originate. A *prototype* typifies a class of designs and serves as a generic design.

### 3.2.4.7 Associative models

The associative mechanism is another key principle of cognition, which is present in design thinking. In associative reasoning concepts are linked on the basis of conceptual relations to form a structure of concepts. This can be represented by a conceptual network, which maps the structure of relationships and emphasise semantics. A semantic network is the set of all relationships which concepts have to other concepts. The semantic network is related to some context in which it has meaning. This provides the basis upon which to model attribute-based associative thinking in design. In typological design there is a restrictive definition of essential formal variables in the type and how they can be hierarchically modelled into a set of formal concepts. In associative reasoning it is the particular structure of conceptual linkages in contrast to a well-defined hierarchical structure, which is significant.

In architectural design, knowledge associated with recognised categories such as building types provides a clear domain example of typological knowledge through which generic designs can be modelled. With regards to design concepts, there is no comparable consensus on what constitutes the vocabulary of architectural concepts. One area where a vocabulary has begun to emerge is that of the *formal concept*. Formal concepts describe particular features (formal attributes) of the design entities. In the case of architecture, these are the vocabulary of concepts, which describe the formal content of building designs. A system like this could allow for the maintenance, presentation and possible modifications of associative linkages between concepts within the designs.

An example of this type of model is the FORMNET system. This system contains a vocabulary of more than hundred formal attributes that are hierarchically organised into nine major categories and 40 sub-categories. These were established through the survey and analysis of the literature on formal analysis and on the architecture of Le Corbusier. The formal knowledge relative to the villas is organised into a semantic network in which the formal attributes are the nodes. This provides a means to navigate within the system by associative connections between formal concepts, to study the coincidence of formal concepts in various designs and to study relationships between attributes. Some of the attributes that are used in FORMNET are symmetry, grid, regulating lines and free plan. The projects are described both two and three-dimensionally, while the concepts are described two-dimensionally. Historical styles such as Doric and Gothic also provide associative models. Doric gives democratic and Gothic religious associations.

### 3.2.4.8 Exemplar models

In this approach it is attempted to re-use prior knowledge rather than to generate new designs. The previous solution is adapted to the current situation. Prior knowledge is associated with specific design cases in which the knowledge is highly explicit.

The case, as specific knowledge, can be distinguished from generic knowledge by its unique departure from the norm. A design case has something specific to communicate regarding the solution, its history of generation and its implications in use. Since the knowledge of prior problem solutions is used, the case is structured in such a way that it can be adapted.

Architectural details are an example of this type of case. Building details are example-based and detailing is often based on the re-use of specific examples, which are exemplars, or examples that function as models. The information base could be very broad and special attention has to be paid to the access method. The traditional CI/SfB indexing conventions are not adequate. Organisation of the index according to a convention of typological categories of elements will support conventional search by taxonomic categories (category, element name and product name). However it will not necessarily support search by other categories such as design principles and it will not support browsing and cross-indexing.

Three broad classes of domain knowledge can be identified:

- *Procedural knowledge* is a process or algorithm for design. The design of a staircase is an example where the calculations are based on floor to floor height, length of the stair run, and the tread riser relationships.
- *Causal knowledge* is a detailed procedure for calculation. An example is the calculation and design of partitions for thermal or acoustic properties.
- *Behavioural knowledge* is the understanding of the performance achieved by particular materials or by a particular configuration of elements in a building. This characterises much of the knowledge of building detailing.

Despite the abundance of literature and information in the field, knowledge is generally poorly structured. The knowledge is not structured in such a way that it can be used in models of the design process. It is the integration of knowledge behind the detail within the working environment, which is a long-term objective of intelligent CAD libraries.

Some desirable characteristics of such a system are:

- Memory and indexing approach to support exploration as well as directed search.
- Explanations such as pitfalls and lessons should be integrated into the case.
- The graphic representation should be linked to a model of the case adaptation process.
- Library and the design environments should be integrated.

### 3.2.4.9 The design precedent

The selection process of relevant ideas from prior designs in current design situations has been termed *precedent-based design*. During the course of exploration of design ideas within precedents, designers are able to browse freely and associatively between multiple precedents in order to make relevant connections. This makes the discovery of unanticipated concepts possible in precedents. In precedent-based systems the ability to encode, search and extract design knowledge relevant to the problem at hand is significant.

One method to represent design knowledge in this type of CBR is to base it upon a decomposition of holistic case knowledge into separate *chunks of design knowledge*. One means to decompose case knowledge into separate and independent chunks is the concept of the story, which is currently employed in the CBR community (Oxman *et al.* 1994:59). The design story is employed as a way to decompose existing descriptions of complex design precedents into chunks. A story is also useful because it provides contextual information. In order to structure a story in a useful format that can be analysed a tri-partite schema that uses an *issue-concept-form* formalism can be used. Each design story is a way to link these three components. Indexing of the cases could become *story indexing* rather than case indexing. Linkages between precedents can be established through matching of issues and design and design concepts.

The design precedent addresses some of the problems of the other models. Because of their network structure, the knowledge representation can use a semantic network or in the form of a node-link structure as provided in hypertext systems.

Precedent-based design is viewed as a significant paradigm in architectural design. However, it has been the subject of less theoretical and research work than typological design. The potential for design aid systems based upon precedent libraries (design thesauri) is another realistic possibility.

## 3.2.5 Case-based Reasoning indexing and retrieval

### 3.2.5.1 Introduction

One of the important issues in CBR is retrieval of appropriate cases. The indexing and retrieval methods as described by Kolodner (1993), Flemming (1994) and Charlton *et al.* (1998) already solved the indexing and retrieval problem substantially. It is clear that the indices required to facilitate the initial selection of relevant cases need to be based on linguistic variables. All present methods are based on static linguistic variables that are searched in order to find the most appropriate case. The author is of the opinion that static linguistic variables fail to address the problem of context of the index. An example of this is a description that states that the design for a specific building is *energy efficient*. The linguistic variable *energy efficient* could have been quantified as a design that requires 50 $W/m^2$. If a significant breakthrough is made in lighting design a new low energy design might be feasible requiring only 20 $W/m^2$. This would invalidate the previous assumption that 50 $W/m^2$ is energy efficient. The best solution to this type of problem is to formulate a dynamic, context sensitive linguistic variable *energy efficient*. The value is calculated at the time of retrieval in terms of the known universe of designs. This implies that it is better to store the calculation method with the linguistic variable, rather than absolute values. In the case of lower order values that are absolute such as *gross area*, *rentable area*, *volume* and *reverberation time* it is acceptable to store the values in an absolute way. If static non-linguistic variables are to be compared and weighed then the Flemming method is convenient to weigh up the various factors. The author is of the opinion that Charlton *et al.* (1998:322) comes the closest to a dynamic approach by recommending the use fuzzy sets. However they fail to recognise the need for dynamic linguistic indices.

### 3.2.5.2 The indexing problem

The *indexing problem* has several parts. When a case is created, appropriate labels must be assigned to ensure that it can be conveniently recalled. Labels are also used at retrieval time to judge the appropriateness of an old case in a new situation. Some of the basic requirements of indices are (Kolodner 1993:194-195):

- They have to anticipate the vocabulary a retriever might use.
- Indexing has to be by concepts that are normally used to describe the items being indexed, whether they are cosmetic features or something more abstract.
- Indexing has to anticipate the circumstances in which a retriever is likely to retrieve something.

Tasks and domains must be analysed to find the functionally relevant descriptors that should be used to describe and index cases. This is called the *indexing vocabulary*. Index vocabulary is a subset of the vocabulary used for full symbolic representations of cases. In the event of retrieval-only CBR it is not necessary to represent the entire contents of the cases symbolically. It is only necessary to represent in the case index the part of the description needed for retrieval. This is called *index assignment*. Indices are those combinations of features of a case that describe the circumstances in which a reasoner might find the case useful during reasoning.

The following general guidelines for choosing indices can be identified (Kolodner 1993:197):

- Indices should be predictive. This is those combinations of descriptors of a case that were responsible for solving it the way it was solved and those combinations that influenced its outcome.
- Predictions that can be made should be useful. They should address the purposes the case will be used for.
- Indices should be abstract enough to make a case useful in a variety of future situations. This often implies that indices should be more abstract than the detail of a particular case.
- Indices should be concrete enough to be easily recognisable in future situations. It should be possible to recognise the case with little inference.

**3.2.5.3 Choosing an indexing vocabulary**

A vocabulary needs to cover relevant similarities rather than just surface features. Focussing the indexing on the relevant features of a case does this. Because indices are chosen from a case's description, the requirements of the indexing vocabulary are known. The case can be described from two main sets of material (Kolodner 1993:203):

1. The *functional approach*. By means of the *functional methodology* representative domain cases are collected. The corpus of available cases and the tasks that must be supported are examined. For each case the points it can make, the situations in which each point is applicable and the ways the case needs to be described to make it available.
2. The *reminding approach*. The kind of reminding that is natural among human experts who do the designated task is examined. Similarities between new situations and the cases they are reminded of. This is an attempt to find out which descriptors are important to judge similarity and the circumstances.

The indexing vocabulary must capture those domain dimensions that are useful for reminding. One of the attempts to a vocabulary for intentional situations was the Universal Index Frame (UIF). In 1982 Schank proposed organising structures called *Thematic Organisational Packets* (TOPs). TOPs are organisers of cases that are thematically similar to each other. If two cases have the same thematic structure they fall into the same thematic category. The *Universal Index Frame* (UIF) of Schank and Osgood in 1990 built on this concept. It uses the dimensions and vocabulary of goal and plan interactions to structure the descriptions of intentional situations. The UIF suggests the following descriptors or dimensions (Kolodner 1993:228):

- *Anticipatory affect:* the emotions of the character going into the situation
- *Pretask belief:* relevant beliefs of the character going into the situation
- *Task:* the task the actor is actively engaged in as the episode plays itself out
- *Theme:* relevant thematic relationships, roles played by the character, character traits and ambitions that the character brings to the situation
- *Goal:* the character's relevant goal
- *Plan:* the plan the character uses or intends to use in the situation
- *Result:* the major impact of what happened in the situation
- *Positive side effects*
- *Negative side effects*
- *Resultant affect:* the emotions of the character leaving the situation
- *Post-task belief:* relevant beliefs of the character leaving the situation. This is what the character learned from the situation
- *Change in affect:* a characterisation of the degree of change in the character's feelings as a result of the episode

### 3.2.5.4 Methods for index selection

Kolodner (1993:249) identified the following general steps of index selection:

1. Determine what the case could be useful for.
2. Determine under what circumstances it would be useful.
3. Translate the circumstances into the vocabulary of the reasoner.
4. Synthesise the circumstances to make them as recognisable and generally applicable as possible.

Kolodner (1993:249-281) provides a detailed description of how indices can be chosen. The description below is a summary of these methods:

1. *Choosing indices by hand.* This is used when the cases are complex and the indices need to be accurate. This is also used when the knowledge required to choose the indices accurately is not concretely available or is too complex to insert directly into the computer.
2. *Choosing indices by machine.* This is useful when the problem solving and understanding are already automated. Three methods of automated index selection exist i.e. *checklist-based, difference-based* and *explanation-based* methods.

*2.1 Choosing indices based on a checklist*

This type of index is based on a specific set of dimensions. The checklist facilitates the process of index selection. For each dimension on the checklist a value is found or computed that describes the case. This method puts a significant responsibility on the system builder, because it is only as good as the previously designed checklist. Typical problems that can be encountered are incomplete checklists that result in insufficient indexing. It is also important to discriminate between important and unimportant dimensions. The following steps summarise the process for setting up a checklist:

- List the tasks that the case retrieval will support.
- For each task, determine the features that tend to predict solutions and outcomes.
- For each kind of feature, compute a set of useful generalisations of the feature. Make sure that the features chosen are recognisable and available during reasoning.
- Create the checklist by collecting the complete set.

The list of heuristics below gives an indication of which features are indexing candidates. Features should chosen that:

- Predict outcomes.
- Be predictive of other features.
- Make the kinds of predictions the reasoner needs.
- Discriminate.

*2.2 Difference-based*

In this indexing method the purpose of indexing is to keep track of the differences between cases. During retrieval search algorithms can choose the best matching cases from the case library. Not all features that are different across cases make useful indices. To ensure that a difference-based index retriever selects only predictive features, difference-based indexing must be combined with some method of choosing predictive features. One way of achieving this is by a combination of *difference-* and *checklist-based* methods.

As discussed above, *checklist-based* indexing methods focus on which dimensions to focus on for indexing. *Difference-based* methods concentrate on which values along any dimension are useful for indexing. A combination of the two methods allows indexing on predictive dimensions that differentiate a case from other similar ones. The following steps summarise the process to set up this type of index:

- Select a classification for each case.
- Select types of features that are known to be predictive. These are usually context sensitive checklists.
- For each feature its value is computed that results in dimension-value pairs.
- From this list all pairs are removed that are non-predictive in the specific context or normative.

The pairs that are left are those that are predictive and that differentiate the case from others.

### *2.3 Explanation-based indexing*

Difference- and checklist-based indexing methods provide a means of computing predictive features to indices. The problem with this is that indices are based on a model of the features that are usually predictive and do not analyse cases individually for their predictive features. This leads to the problem that features are selected that are not predictive for the particular case or features that are predictive are not indexed.

Explanation-based indexing methods attempt to choose indices appropriately for individual cases. The reasoner uses explanation-based generalisation methods to generalise the explanation. Indices are then chosen from the content of the generalised explanation. In explanation-based indexing, domain knowledge is used to determine which facts of a case are relevant and which can be safely ignored. The index is generalised to the most abstract point where the explanation can still hold. After the reasoner discovers it has made a mistake, it attempts to explain it or assign blame. After explaining the mistake, it extracts from the explanation the concrete recognisable features of the situation that that were responsible for the problem. It then generalises those features to the point where they are still concrete but where the explanation that was derived can still be applied. Unlike checklist-based methods this method chooses as indices only those features that are responsible for the failure. Those features, if observed in future cases, will predict the failure observed in this case. Checklist- and difference-based methods have no means of distinguishing which of the many potentially predictive and differentiating features are responsible for the failure, and will index using far more features.

The explanation-based index selection process consists of the following steps:

1. Create an explanation.
2. Select relevant observable features from the explanation.
3. Generalise those observable features as far as possible, making sure the resulting generalisations are also observable. The original explanation must still apply, given this general description.
4. If the index supports a solution-creation goal, then
   - append additional information specifying the goal the case achieves.
   - generalise the goal appropriately and repeat the process.

### 3.2.5.5 Retrieving cases from the case library

If a large case design library has been built up it should be possible to conveniently retrieve a case by means of a retrieval procedure. Flemming (1994:84) identified a method that uses the principle of a target index, $t$. $t$ is compared with all available cases $c$. Given a target index, the comparison with a case index proceeds object-by-object and attribute-by-attribute for those objects that belong to and those attributes that have values in the target index. In the simplest case, only objects of the same class or type and attributes with the same names are compared. In order to extend the allowed matches several schemes are used that include subtype, subrange and subset matching. The following general rules are defined:

- An object *A* comparable to an object *B* if *B* belongs to the same class or to a subclass of *A*.
- An attribute *a* is comparable to an attribute *b* if *a* is implied by *b*.

Comparability of attributes implies that attributes have values of the same type. For example a minimum x-dimension attribute is implied by a minimum dimension attribute. In the simplest case, the results of comparisons are binary. Sometimes it is important to distinguish whether a lower bound is missed narrowly or by a large degree. It may also be important to know whether only one or several functional units are missing when two *constituent* attributes are compared. Flemming (1994:85) suggests that for each comparison the degree to which it succeeds be computed. This is expressed as the *closed bounded interval* [0,100]. 100 is a perfect match and lower numbers indicate the percentage by which a perfect match has been missed.

When scanning cases for retrieval the individual comparisons must be aggregated so that cases with the best overall fit are presented to the designer. One solution to arrive at ranked values of cases would be to compare the weighed sums of the individual matches. However plausible weights are difficult to determine in building design. Interactions between comparisons cannot be taken into account. The way in which certain deficiencies enter an overall evaluation may depend crucially on the way other comparisons succeed. This is known as the problem of *mutual preferential dependence*. To avoid some of these problems problem features are divided into predetermined priority classes and matches for prioritised features are determined first. Cases that match the most features are preferred. This process uses weighed sums implicitly. Features in the same class are given the same weight and matches are added up. Features in higher classes overrule those in lower classes. The designer is able to decide which features he is going to search on.

The method of calculation can be formalised in the following way by defining a special retrieval function

$$\varphi(t,c)$$

which returns a real number in the *closed bounded interval* [0,100] to express the match between a target index $t$ and a case index $c$.

$\varphi$ unpacks $t$ recursively in terms of its valued attributes and computes their matches with comparable valued attributes in $c$.

$\varphi$ is specialised with regards the objects or data types that have to be compared. Some special forms of $\varphi$ are indicated below. The subscripts indicate the type of specialisation.

$$\varphi_{OBJ}(A,B) = \begin{cases} 0 & \text{if } A \text{ and } B \text{ are not comparable;} \\ 100 & \text{if } A \text{ has no valued attribute;} \\ \sum_a w_a \varphi_{ATTR,OBJ}(a,B) & \text{otherwise,} \end{cases}$$

Unordered lists like the value of attribute tests can be compared similarly to $\varphi_{OBJ}$. The sum goes over all valued attributes $a$ of $A$. $w_a$ are weights with

$\sum_a w_a = 1$; and

$$\varphi_{ATTR,OBJ}(a,B) = \begin{cases} \varphi_{ATTR}(a,b) \text{ if } B \text{ contains a valued attribute } b \text{ comparable with } a\text{;} \\ 0 \text{ otherwise} \end{cases}$$

$\varphi_{ATTR}(a,b)$ is specialised with respect to the data type of $a$ and $b$. For attributes whose data types are lower bounds (like the attributes *minimum-width* and *min-area* of a functional unit of variable size.

$$\varphi_{LOWER\_BOUND}(a,b) = \begin{cases} 100\ [b]/[a] & \text{if } [b] < [a]; \\ 100 \text{ otherwise.} \end{cases}$$

Where $[x]$ denotes the value of an attribute $x$. Upper bounds as well as general numbers can be treated similarly. Names match either completely (100) or not at all (0).

The basic form of $\varphi$ that unravels a *constituent* attribute in a target index can be defined as follows:

$$\varphi_{CONST\_LIST}(K,L) = \max \sum_k w_k \varphi_{OBJ}[k, \gamma(k,L)]$$

where the sum goes over all objects $k$ in $K$. $\gamma(k,L)$ is defined as an operator that traverses $L$ and the constituents of the objects in $L$ to grab a corresponding object comparable with $k$. $\gamma$ satisfies the following conditions:

1. If no corresponding object can be found $\gamma$ returns a dummy object to enforce a 0 value of $\varphi$ for this particular $k$.
2. Repeated calls to $\gamma$ will not return objects that have been returned before; that is, the mapping established by $\gamma$ from objects in $K$ to objects in $L$ and their constituents is right-unique.
3. If $\gamma$ maps an object in $K$ to a constituent $m$ of an object $l$ in $L$, it does not map other objects in $K$ to objects on the path from $l$ to $m$.

Charlton *et al.* (1998:324) state that the descriptions on which retrieval of a relevant case depends are ultimately based on classifications. A classification consisting of restricted values is seen as a flat classification. The very reason for the existence of a classification is to enable stored cases to be retrieved. The methodology discussed above is not very useful at the level where the decision must taken if a design case is appropriate at all for the design problem under consideration. This methodology is more appropriate at a direct technical level where various different technical factors need to be directly considered before a final conclusion is reached. It is difficult to develop suitable indices, because it needs to consider the reference framework of the user. Static indices are rigid because they are unable to adapt to the context of use.

Charlton *et al.* (1998:322) suggested the use of static fuzzy sets with labels that are meaningful to the designer. For each prototypical case, designers are asked to specify its membership values in fuzzy sets. Each label specifies the degree to which the particular artefact is part of the fuzzy set identified by the label's name. Collectively, the labels can be seen as providing a multitude of descriptive names for a case, instead of a single possibility. The use of fuzzy sets as described by Charlton is more flexible than Flemming's, however the membership values in the fuzzy sets themselves are still static. The author is of the opinion that this can be significantly improved by means of a method of dynamic fuzzy sets. A description of this proposed methodology follows below.

### 3.2.5.6 The use of fuzzy sets for case indexing

It is now 35 years since the first creation of Fuzzy Sets and Fuzzy Logic that bridge mathematical precision and the vagueness of common-sense reasoning. Fuzzy sets have been successfully implemented in numerous commercial products such as vacuum cleaners, washing machines, rice cookers and cameras that resulted in energy efficiency and increased convenience for the consumer. The Japanese city of Sendai has been using subway trains controlled by fuzzy logic since 1986.

Bellman and Zadeh (1970) and Bojadziev *et al.* (1995:113) describe fuzzy sets as a special class of object in which there is no sharp boundary between those objects that belong to the class and those that do not. Below follows a short summary of the main characteristics of fuzzy sets.

(1) Let $A = \{(x, \mu_A(x)) \quad | \quad x \in X, \mu_A(x) \in [0,1]\}$

Where $\mu_A(x)$ is a function called the *membership function* of *x in A*. $\mu_A(x)$ and $\mu_A : X \rightarrow M$ is a function from $X$ to a space called the membership space. When $M$ only contains two points, 0 and 1, $A$ is nonfuzzy and its membership is identical with the characteristic function of a nonfuzzy set. It can be assumed that $M$ is the closed interval [0,1], with 0 and 1 representing respectively the lowest and highest grades of membership.

A fuzzy set is *normalised* when at least one $x \in X$ attains the maximum membership grade 1, otherwise the set is called *non-normalised*. Assume that the set *X is non-normalized,* then *max* $\mu_A(x) < 1$. To normalise the set *X* means to normalise its membership function $\mu_A(x)$. This is given by:

$$\frac{\mu_A(x)}{\max \mu_A(x)}$$

*Empty set. A* is called an empty set labelled $\phi$ if $\mu_A(x) = 0$ for each $x \in A$.

*Fuzzy singleton*. The fuzzy set $A = \{(x_i, \mu_A(x_i))\}$, where $x_i$ is the only value in $A \subset U$ and $\mu_A(x_i) \in [0,1]$.

$\alpha$ *- level set* or $\alpha$ *-cut.* This is denoted by $A_\alpha$ and is the crisp set of elements which belong to $A$ at least to the degree $\alpha$ :
$A_\alpha = \{x \mid x \in U, \mu_A(x) \geq \alpha\}, \alpha \in [0,1]$

*Strong* $\alpha$ *- level set.* This is defined by:
$A_\alpha^{'} = \{x \mid x \in U, \mu_A(x) > \alpha, \}, \alpha \in [0,1]$

In many practical situations the membership function $\mu_A$ has to be estimated from partial information about the subject of consideration. The problem of estimating $\mu_A$ from the knowledge of the set of pairs $(x_1, \mu_A(x_1)),....,(x_N, \mu_A(x_N))$ is the problem of abstraction. This problem plays a central role in pattern recognition, but also in the selection of a suitable case for architectural design. Similar abstractions had to be made to calculate the average condition and suitability of facilities during the National Health Facilities Audit (NHFA) in South Africa. In the case of the NHFA criteria had to be carefully derived to determine what the rating of a particular construction element should be on a scale of [1,5] where 5 denoted the optimum and 1 the worst case scenario. In the internal calculations this was replaced by a normalised *closed bounded interval* [0,1].

*Equality.* Two fuzzy sets are equal, written as $A = B$, if and only if $\mu_A = \mu_B$. That is $\mu_A(x) = \mu_B(x)$ for all $x$ in $X$ .

*Containment.* A fuzzy set $A$ is contained in or is a subset of fuzzy set $B$, written as $A \subset B$, if and only if $\mu_A \leq \mu_B$. The fuzzy set of energy efficient buildings is a subset of the fuzzy set of buildings.

*Complementation.* $A'$ is said to be the complement of $A$ if and only if $\mu_A' = 1 - \mu_A$. For example, the fuzzy sets: $A = \{\, high\_rise\_buildings \,\}$ and $A' = \{\, not\ high\_rise\_buildings \,\}$ are complements of one another if the negation "not" is interpreted as an operation which replaces $\mu_A(x)$ with $1 - \mu_A(x)$ for each $x$ in $X$.

*Intersection.* The intersection of $A$ and $B$ is denoted by $A \cap B$ and is defined as the largest fuzzy set contained in both $A$ and $B$. The membership function of $A \cap B$ is given by

(2) $\mu_{A \cap B}(x) = Min(\mu_A(x), \mu_B(x)), x \in X$ where $Min(a,b) = a$ if $a \leq b$ and $Min(a,b) = b$ if $a > b$. In infix form, using the conjunction symbol $\varpi$ in place of $Min$, (2) can be written more simply as

(3) $\mu_{A \cap B} = \mu_A \varpi \mu_B$. The notion of intersection bears a close relation to the connective "and". If $A$ is the class of high rise buildings and $B$ is the class of energy efficient buildings, then $A \cap B$ is the class of buildings that is both high rise and energy efficient. It should be noted that in the example "and" is interpreted in a "hard" sense. That is, we do not allow any trade-off between $\mu_A(x)$ and $\mu_B(x)$ so long as $\mu_A(x) > \mu_B(x)$ or vice-versa. For example if $\mu_A(x) = 0.8$ and $\mu_B(x) = 0.5$, then $\mu_{A \cap B}(x) = 0.5$ so long as $\mu_A(x) \geq 0.5$. In some cases, a softer interpretation of "and" which corresponds to forming the algebraic product of $\mu_A(x)$ and $\mu_B(x)$, rather than the conjunction $\mu_A(x) \varpi \mu_B(x)$ may be closer to the intended meaning of "and". From the mathematical as well as the practical point of view, the identification

of "and" with ϖ is preferable to its identification with the product, except where ϖ clearly does not express the sense in which one wants "and" to be interpreted.

*Union.* The union of $A$ and $B$ is denoted by $A \bigcup B$ and is defined as the smallest fuzzy set containing both $A$ and $B$. The membership function of $A \bigcup B$ is given by

(4) $\mu_{A \cup B}(x) = Max(\mu_A(x), \mu_B(x)), x \in X$ where $Max(a,b) = a$ if $a \geq b$ and $Max(a,b) = b$ if $a < b$. In infix form, using the disjunction symbol ω in place of $Max$, (4) can be written more simply as

(5) $\mu_{A \cup B} = \mu_A \, \omega \, \mu_B$.

As in the case of the intersection, the union of $A$ and $B$ bears a close relation to the connective "or". If $A$ = {*high_rise_buildings*} and $B$ = {*energy_efficient_buildings*}, then $A \bigcup B$ = {*high_rise_buildings* or *energy_efficient_buildings*}. As in the case mentioned above a "hard", "or" which corresponds to (5) and a soft "or" that corresponds to the algebraic sum of $A$ and $B$ can be distinguished. This latter is denoted by $A \oplus B$ and is defined by (7).

*Algebraic product.* The algebraic product of $A$ and $B$ is denoted by $AB$ and is defined by

(6) $\mu_{AB}(x) = \mu_A(x)\mu_B(x), x \in X$.

*Algebraic sum.* The algebraic sum of $A$ and $B$ is denoted by $A \oplus B$ and is defined by

(7) $\mu_{A \oplus B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x), x \in X$.

From (7) it follows that

(8) $A \oplus B = (A' B')'$.

ϖ and ω are associative and distributive over one another. (product) and ⊕ (sum) are associative but not distributive.

### 3.2.5.7 The use of fuzzy sets to formulate dynamic linguistic variables for case retrieval

Variables whose values are words or sentences in natural or artificial language are called *linguistic variables*. Natural language words is a convenient means to retrieve architectural design cases, because humans think in terms of words that most closely describe the desired design qualities. To illustrate the concept of a *linguistic variable* consider the word *age* in a natural language. The meaning of this word is the summary of an enormous large number of individuals. It cannot be characterised precisely. This word also has a different meaning in different domains. The meaning of *age* in a building domain is something totally different to *age* in a human context. The discussion will continue with *age* in the context of buildings. By means of fuzzy sets *age* can be described more precisely. *Age* is a linguistic variable consisting of fuzzy sets such as *very_new*, *new*, *old* and *historic*. These words are called terms of the linguistic variable *age*. Each term is defined by an appropriate membership function. Bojadziev (1995:178) states that good candidates for membership functions are triangular, trapezoidal or bell-type shapes with or without a flat. These mathematical shapes describe the different ways membership functions can be structured. An example is a triangular fuzzy number that are very often used in applications such as fuzzy controllers, managerial decision making and the social sciences. The underlying advantage of the fuzzy relationship shapes mentioned is that membership functions for terms using them can be constructed on the basis of little information.

Let us describe the linguistic variable *age* on the universal set $U = [0,200]$ (Figure 14) by means of triangular fuzzy numbers, which specify the terms *very_new*, *new*, *old* and *historic*.

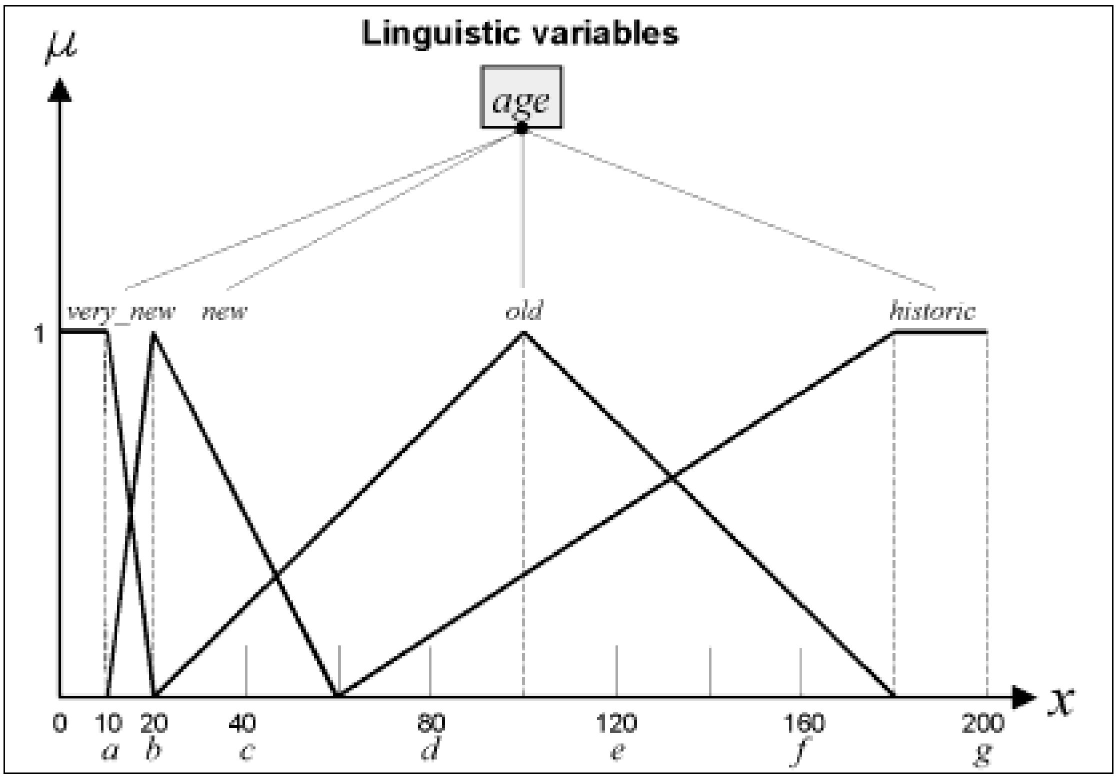


Figure 14: Terms of the linguistic variable *age* in a building context (Author)

The membership functions of the terms using a triangular calculation are:

$$\mu_{very\_new}(x) = \begin{cases} 1 & \text{for} \quad 0 \le x \le 10, \\ \dfrac{20-x}{10} & \text{for} \quad 10 \le x \le 20, \end{cases}$$

$$\mu_{new}(x) = \begin{cases} \dfrac{x-10}{10} & \text{for} \quad 10 \le x \le 20, \\ \dfrac{60-x}{40} & \text{for} \quad 20 \le x \le 60, \end{cases}$$

$$\mu_{old}(x) = \begin{cases} \dfrac{x-20}{80} & \text{for} \quad 20 \le x \le 100, \\ \dfrac{180-x}{80} & \text{for} \quad 100 \le x \le 180, \end{cases}$$

$$\mu_{historic}(x) = \begin{cases} \dfrac{x-60}{120} & \text{for} \quad 60 \le x \le 180, \\ 1 & \text{for} \quad 180 \le x \le 200, \end{cases}$$

Note that the triangular $\mu$ values for the linguistic terms of *age* are not linear. In this case it can also be seen that compression of scale occurs at the *very_new* end. An important limitation, of linguistic terms defined like these in the example, is that they are static. This implies that the structure is not self-adjusting if the context where the terms that are used changes. This limits the universal application of terms that are defined in this way. The author proposes a system of linguistic variables to be defined that stores the calculation method. When the linguistic variable is brought into a specific context, then the terms would assume the correct relative values in the context of the specific environment.

Linguistic variables are important in applications. The parameters of technical systems such as condition, suitability, energy, temperature, weight, speed, pressure and heat can be understood as linguistic variables.

### 3.2.5.8 Fuzzy set linguistic modifiers

Let $x \in U$ and $A$ is a fuzzy set with membership function $\mu_A(x)$. Assume that $m$ is a *linguistic modifier* such as *very, not* and *fairly*. $mA$ is a modified fuzzy set whose membership function $\mu_{mA}(x)$ is a composition of a suitable function $f(x)$ and $f(\mu_A(x))$.

The following selections for $f(x)$ are often used to describe the modifiers *not, very* and *fairly*.

| | | |
|---|---|---|
| $f(x) = 1 - x$ | not, | $\mu_{notA}(x) = 1 - \mu_A(x),$ |
| $f(x) = x^2$ | very, | $\mu_{veryA}(x) = [\mu_A(x)]^2,$ |
| $f(x) = x^{\frac{1}{2}}$ | fairly, | $\mu_{fairlyA}(x) = [\mu_A(x)]^{\frac{1}{2}}.$ |

Consider the fuzzy set $A$ that describes the size of a particular facility in terms of gross m² by the linguistic variable *SIZE*. Assume a small database of five facilities each having a specific gross m².

| Facility Name | Facility Code | Gross m² |
|---|---|---|
| Facility 1 | $F_1$ | 5 830 |
| Facility 2 | $F_2$ | 1 431 |
| Facility 3 | $F_3$ | 12 979 |
| Facility 4 | $F_4$ | 11 500 |
| Facility 5 | $F_5$ | 7 500 |

Assume further that *SIZE* has three terms *large*, *medium* and *small* having the following values in the *closed bounded interval* [0,1].

| | | |
|---|---|---|
| *large* | [0.66,1] | $0.66 \leq x \leq 1$ |
| *medium* | [0.33,0.66) | $0.33 \leq x < 0.66$ |
| *small* | [0,0.33) | $0 \leq x < 0.33$ |

In order to ensure a dynamic and flexible system the values of the terms are expressed in terms of a universal set in the context of the application under consideration. The gross m² values therefore range from $[Min(F_{area}), Max(F_{area})]$ where $F_{area}$ is the facility gross area.

In terms of the small example database above the range of values from *small* to *large* would be:

(1) [1431,12979]

Assume that Facility $F_x$ has a gross m² area of 8 000 m². In terms of the data above it can be stated that:

(2) $$\mu_{size}(x) = \frac{F_{x\_area}}{(Max(F_a) - Min(F_a))} = 0{,}69 = large$$

$F_{x\_area}$ is the area of Facility *x*.

Assume that a new facility with a gross area of 15 000 m² is added to the database above. In terms of Facility $F_x$ the following is now true:

(3) $$\mu_{size}(x) = \frac{F_{x\_area}}{(Max(F_a) - Min(F_a))} = 0{,}59 = medium$$

Due to the inclusion of the large facility, $F_x$ has been reclassified as *medium*. Due to the flexible formulated definitions the system under consideration will be self-adjusting. This is especially useful when fuzzy sets are considered that give a measure of performance such as energy use.

In the case of interpreting case indices the *intersection*, *union* and *complement* are the most useful. The author is of the opinion that fuzzy sets can be used to select the most appropriate cases from the possible set of cases in the CBR based system envisioned. If a vocabulary of words can be carefully selected that best describe certain index phenomena then, by means of a process of abstraction suitable $\mu_A$, values can be allocated.

The author comes to the conclusion that the calculations that were made in the NHFA with regards condition and suitability are really a subset of total number of possible fuzzy sets possible in this domain. To quantify condition in abovementioned audit the author allocated discrete meanings to fuzzy condition rating words such as *as_new*, *maintain*, *repair*, *replace/ upgrade* and *condemn/ leave*. Each of these keywords was allocated a value in the [0,1] range:

| | |
|---|---|
| *as new* | *= 1,0* |
| *maintain* | *= 0,8* |
| *repair* | *= 0,6* |
| *replace/ upgrade* | *= 0,4* |
| *condemn/ leave* | *= 0,2* |

In a similar way suitability assessments were allocated keywords with values in the [0,1] range:

| | |
|---|---|
| *ideal* | *= 1,0* |
| *acceptable* | *= 0,8* |
| *tolerable* | *= 0,6* |
| *hardly tolerable* | *= 0,4* |
| *intolerable* | *= 0,2* |

The contribution of cost per gross m² for each of 98 construction elements was derived from an analysis of bills of quantities from quantity surveyors. During the audit the average condition for a particular facility was derived by means of the following:

(1) $\sum_{e=1}^{98} w_e ca$ where the meaning of the variables is:

$e =$ *the construction element number*

$w_e =$ *the condition cost model weight in the range* $[0,1]$

$c =$ *condition rating in the range* $[0,1]$

$a =$ *gross area in m² of the department, building or total facility are where the element occurs.*

In a similar way the average suitability for the facility was calculated as:

(2) $\sum_{e=1}^{98} w_e sa$ where the meaning of the variables is:

$e =$ *the construction element number*

$w_e =$ *the suitability cost model weight in the range* $[0,1]$

$s =$ *suitability rating in the range* $[0,1]$

$a =$ *gross area in m² of the department, building or total facility are where the element occurs.*

Abovementioned calculations resulted in the average condition and suitability per facility that could be summarised to district, region, province and country level. The conclusion is made that on the basis of abovementioned concept super concepts can be defined that would facilitate the powerful manipulation of derived high level concepts. The processing domain of the facilities audit is limited. This makes it feasible to formulate fuzzy data abstractions. The keyword *SIZE* can be expressed in terms of the size in gross *m²* found in the audit. It is impossible to define *SIZE* as a universal quantified concept.

An example of an expression in fuzzy terms could be the following:

List all *large* hospitals (size description), in a *new* condition that occur in the province of *Western Cape* (location). In this case we have two fuzzy variables and one non-fuzzy variable. If suitable ranges of words (labels) can be defined and by means of abstraction fuzzy values be allocated then the fuzzy operations, *intersection* and *union* can be used.

Structured Query Language (SQL) as implemented in Oracle has the capability to process sets, although this capability is not often used. The following SQL operators are available in the SELECT statement:

| | |
|---|---|
| UNION | *Combines two queries and returns all distinct rows returned by either individual query.* |
| UNION ALL | *Combines two queries and returns all rows returned by either query, including duplicates.* |
| INTERSECT | *Combines two queries and returns all distinct rows returned by both individual queries.* |
| MINUS | *Combines two queries and returns all distinct rows by the first but not by the second.* |

These set operators make it possible to implement traditional set theory easily. This can readily be expanded to implement fuzzy sets. The following would be required:

- Definition library (labels) of concepts that will typically be manipulated.
- Standard functions and procedures that can be included in a CBR program or object.
- A data set where the domain operational parameters are known.

The following list of linguistic fuzzy sets and terms can be defined for use in queries related to the life cycle of buildings:

*CONDITION*

| | |
|---|---|
| *new* | *= 1,0* |
| *maintain* | *= 0,8* |
| *repair* | *= 0,6* |
| *replace* | *= 0,4* |
| *condemn* | *= 0,2* |

*SUITABILITY*

| | |
|---|---|
| *ideal* | *= 1,0* |
| *acceptable* | *= 0,8* |
| *tolerable* | *= 0,6* |
| *hardly tolerable* | *= 0,4* |
| *intolerable* | *= 0,2* |

*SIZE*

*very large*
*large*
*average*
*small*
*very small*

*AGE*

*very old*
*old*
*recent*
*new*

*DISTANCE*

*far*
*close*

*LARGE*

*exceptionally large*
*very large*
*large*
*average size*

*SMALL*

*very small*
*small*

*average size*

*UTILISATION*

*totally over utilised*
*very over utilised*
*over utilised*
*normal use*
*under utilised*
*significantly under utilised*
*under utilised*

In all cases the keyword that is closest to the main subject in the list appears at the top of the list. These keywords can be converted into static fuzzy set labels by allocating approximate membership values. It can be assumed that $\mu_A$ will be in the range [0,1]. In the case of CONDITION the values could be:

| | |
|---|---|
| *new* | $0.9 \leq \mu_A(x) \leq 1.0$ |
| *maintain* | $0.7 \leq \mu_A(x) < 0.9$ |
| *repair* | $0.5 \leq \mu_A(x) < 0.7$ |
| *replace* | $0.3 \leq \mu_A(x) < 0.5$ |
| *condemn* | $0.0 \leq \mu_A(x) < 0.3$ |

In this case all the values are linear. The abstraction to the particular values was calculated in such a way as to get a clear distinction between the different condition categories in order to map it to colours. In this case absolute accuracy was not important. It doesn’t matter how many terms the particular fuzzy set contains. If the fuzzy set only contains two categories then it becomes a traditional set.

If a user needs to define a dynamic fuzzy set that displays a list of all *new* (condition) buildings, that is *small* (gross area) the equivalent dynamic database SQL statement could be the following:

```
SELECT FacilityCode
FROM FacilityResource
WHERE (RemainingResource/TotalResource) >= 0.9
AND ConditionCode = 0

INTERSECT

SELECT FacilityCode
FROM Facility
WHERE ((FacilityArea/(MAX(FacilityArea) – MIN(FacilityArea)) < 0.4)
AND ((FacilityArea/321300.0) >=0.2)

INTERSECT

SELECT FacilityCode
FROM Facility
WHERE FacilityCode LIKE ‘WCP%’;
```

### 3.2.6 Conclusion

Kolodner (1993:263) suggests that the following methods be used to maintain context sensitivity in the case index selection.

- Use several checklists, each organised around a different well-known context.
- Keep track of how useful individual indices are and modify lists when they are not useful.

Kolodner (1993) also suggested the method of parameter adjustment for interpolating values in a new solution based on those from an old one. In parameter adjustment changes in parameters in an old solution are made in response to differences between problem specifications in an old and a new case. Several case-based reasoning systems use parameter adjustment as a method of adaptation. A system called PERSUADER adjusts old labour-management contracts with new information. If an old contract was signed in a location where the cost of living is high and has risen faster than the norm, but it is not the case in the new dispute, then a smaller percentage wage increase is in order in the new contract.

In all cases the use of a dynamic adaptive fuzzy set based indexing system comes the closest in solving the problems associated with context sensitivity and parameter adjustment. The inherent flexibility of fuzzy sets make them ideal for indexing in many different environments as well as level of detail. This will be the case with design cases found in the construction industry.

Linguistic variables offer a convenient means to intensify or soften the effect of the terms of a fuzzy set.

## 3.3 The systems view of the world

### 3.3.1 Introduction

In this section manufacturing, concurrent engineering, Taguchi techniques and the Fuzzy Front End is included because of the prominence of these in the world of manufacturing. Architectural design is seen as a type of low-quantity production. Concurrent Engineering attempts to speed up the engineering process in order to be more effective. Taguchi Techniques indicate how big the impact of small variations in critical parameters or dimensions might be. The Fuzzy Front End provides an opportunity to buy value time during the design process.

In this study it is proposed that architectural design experience be packaged in cases and design starter kits. A case is seen as an entire project where all the design knowledge is stored in the form of artefact descriptions and process descriptions. Artefact descriptions consist of shape- and functional views. The process description consists of sequences over time. The author observed that the data that are required to structure the existence of an artefact over the life cycle exist in different worlds spread over time. These worlds were identified in chapter 3.2. It is observed that attempts to unify the different worlds into one single model such as the Industry Foundation Classes, discussed under 3.2 is unlikely to succeed. In the author's experience it is far more flexible to use processes as a means of formulating relationships between these worlds over time. This has been tested in the PREMIS facilities management system.

The present study attempts to store architectural design knowledge in the form of cases and to create portable mini architectural design cases (starter kits) which many domain-specific tools such as CAD and spreadsheets can share. The term mini design case refers to a design case at a level where it becomes portable and small enough to plug into many different design environments. It must also be small enough to be conveniently distributable via the World Wide Web. This approach is supported by Charlton *et al.* (1988:311) where a Common Product Data Model (CPDM) is mentioned. In the CPDM design data is represented by structured objects, which can be shared. The CPDM can capture a large portion of the data involved in the artefact's development without coercing artefacts into static class hierarchies. This allows flexible and dynamic modelling in terms of multiple object perspectives, dynamic object reclassification and dynamic class evolution. Attempts to achieve this were made in the prototype system *AEDES*, however the integration between the artefact description and process description is still primitive.

Once a user has decided to use a specific mini design case it will be brought into a specific context. The purpose of this chapter is to explore the fundamental characteristics of the context of the project environment that consists of main topics such as processes, product modelling and life cycle decision validation. In a highly competitive environment the processes and product modelling could be concurrent. Although there are strong similarities between the construction and manufacturing industry there are also fundamental differences.

It is important to realise that the best techniques would not succeed if the motivation, attitude, spirit, personality are not supported by the corporate culture.

### 3.3.2 What is manufacturing?

The word *manufacture* is derived from two Latin words *manus* (hand) and *factus* (made). The combination means made by hand. Modern manufacturing is accomplished by automated and computer-controlled machinery that is manually supervised. Manufacturing can be defined in many ways with two directly applicable to the manufacturing industry (Groover 1996):

- Technologically.
- Economically.

Other types of manufacturing not addressed in this study are:

- Energy manufacturing.
- Environmentally.
- Informatically.
- Socially.

Technologically, manufacturing is the application of physical and chemical processes to alter the geometry, properties and appearance of a given starting material to make parts or products. The processes to accomplish manufacturing involve a combination of machinery, tools, power and manual labour (Figure 15). Manufacturing is almost always carried out as a sequence of operations. Each operation brings the material closer to the final desired state.

Economically, manufacturing is the transformation of materials into items of greater value by means of one or more processing and/or assembly operations (Figure 15). Manufacturing adds value to the material by changing its shape or properties or by combining it with other materials that have been similarly altered.

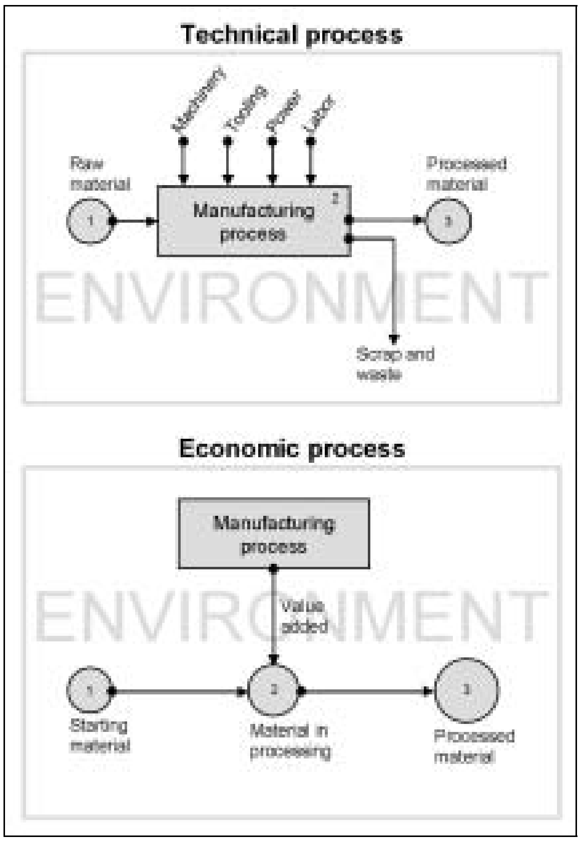


Figure 15: Two ways to define manufacturing, a technical or an economic process (Groover 1996:3)

Groover (1996) identifies *primary*, *secondary* and *tertiary* industries. *Primary industries* are those that cultivate and exploit natural resources such as agriculture and mining. *Secondary industries* take the outputs of the primary industries and convert them into consumer and capital goods. Manufacturing is the principal activity in this category, but it also includes construction and power utilities. *Tertiary industries* constitute the service sector of the economy.

The quantity of products made by a factory has an important influence on the way its people, facilities and procedures are organised. Production quantity refers to the number of units produced annually of a particular product type. Product variety refers to different product designs or types that are produced in the plant. The construction industry is presently a low quantity high variety industry. There is an inverse correlation between product variety and production quantity in terms of factory operations. If a factory's product variety is high, then its production quantity is likely to be low. If the production quantity is high, then product variety will be low. The terms soft and hard product variety can be identified. *Soft product variety* occurs when there are only small differences between products, such as the differences between car models made on the same production line. In an assembled product, soft variety is characterised by a high proportion of common parts among the models. In hard product variety, the products differ substantially and there are few common parts, if any. Again the construction industry is unique in the sense that a lot of parts are common at a low level, but a large variety exist at higher levels. There is also variation between the various construction trades as to the level of standardisation that can be achieved. Air-conditioning parts can be standardised and pre-assembled in a factory before being brought onto site, however this is less feasible with structural elements such as slabs, columns and walls.

#### 3.3.2.1 Manufacturing capability

Manufacturing plants consist of *processes* and *systems* designed to transform a certain limited range of *materials* into products of increased value. The three building blocks, materials, processes and systems constitute the subject of modern manufacturing. There is a strong interdependence among these factors. A company engaged in manufacturing cannot do everything. *Manufacturing capability* refers to the technical and physical limitations of a manufacturing firm and each of its plants. The following dimensions of this capability can be identified:

- Technological processing capability.
- Physical size and weight of product.
- Production capacity.

#### 3.3.2.2 Manufacturing processes

Manufacturing processes can be divided into two basic types:

- Processing operations.
- Assembly operations.

A processing operation transforms a work material from one state of completion to a more advanced state that is closer to the final desired product. It adds value by changing the geometry, properties or appearance of the starting material. An assembly operation joins two or more components in order to create a new entity, which is called an assembly or sub-assembly.

#### 3.3.2.3 Low-quantity production

Groover (1996:21) describes this type of production as a low-quantity range of 1 to 100 units/year. The construction industry bears a close resemblance to this type of manufacturing. In manufacturing the term job shop is often used to describe this type of production facility.

A job shop makes low quantities of specialised and customised products. The products are typically complex, such as space capsules, prototype aircraft and special machinery. Construction activities are normally not nearly as complex as the former.

A job shop must be designed for maximum flexibility in order to deal with the wide product variations encountered. In an analysis by the author of a large construction project this is evident in the large variation of project team configurations and types of construction projects undertaken. If the product is large and heavy and difficult to move in the factory, it typically remains in a single location during its fabrication or assembly. Workers and processing equipment are brought to the product, rather than moving the product to the equipment. Examples of such products include ships, aircraft, railway locomotives and heavy machinery. These products are usually built in large modules at single locations and then the completed modules are brought together for final assembly using large-capacity cranes. In South Africa these practices are not widespread in the construction industry and in-situ construction predominates.

### 3.3.3 Concurrent engineering (CE)

Many terms have been used to describe similar approaches, including simultaneous engineering, life cycle engineering, design integrated manufacturing, design fusion, early manufacturing involvement, parallel engineering, concurrent design and design in the large.

Ziemke *et al*. (1993:26) trace the origins of CE back to 1940, during the Second World War. The American Aviation Corporation received an order for 320 NA-73 fighter aircraft from the British Air Purchasing Commission. These aircraft were later known as the US P-51 Mustangs. The condition of this order was that the first prototype, NA-73X, had to be ready for testing 120 days after receipt of contract. Given the short schedule one would have assumed that the design engineers would only have used proven and conservative design features. Instead the Mustang included the first use of novel concepts such as laminar flow airfoils and the introduction of a combined radiator housing-ejector nozzle that provided 300 pounds of jet thrust, instead of the usual radiator air drag. The aircraft was designed and built in 102 days. During that time, 2 800 drawings representing 600 000 hours of effort were produced. In retrospect it now seems that critical success factors in such wartime design and development teams were their small size, their broadly experienced leadership and above all, motivation.

Currently there is a different reason for CE. During the last decade the life cycle time of products from different branches of industry decreased while the time spent on product development greatly increased. This is known as the time-trap. In terms of the local construction industry this is an over-simplification, because other factors such as the period of high inflation and the fact that buildings are still constructed for a relatively long life. In the new Menlyn Shopping centre project the planning horizon is 25 years. Due to these changes the pay-off period between market entry and amortisation extended as well. In order to meet the challenges of successfully competing in innovative markets the development and design of new products has become one of the most significant factors. The situation can be characterised by three main tendencies:

- Shift from a seller’s to a buyer’s market.
- Increasing globalisation.
- Change in the importance of technology.

The optimisation of the magic triangle that consists of time, quality and costs is necessary to face competition and complexity in the changed environment described above.

The most important elements when applying CE are people and the design of product development processes. Co-operation and communication are regarded as the most important success factors by companies, which are successfully practising CE. This involves:

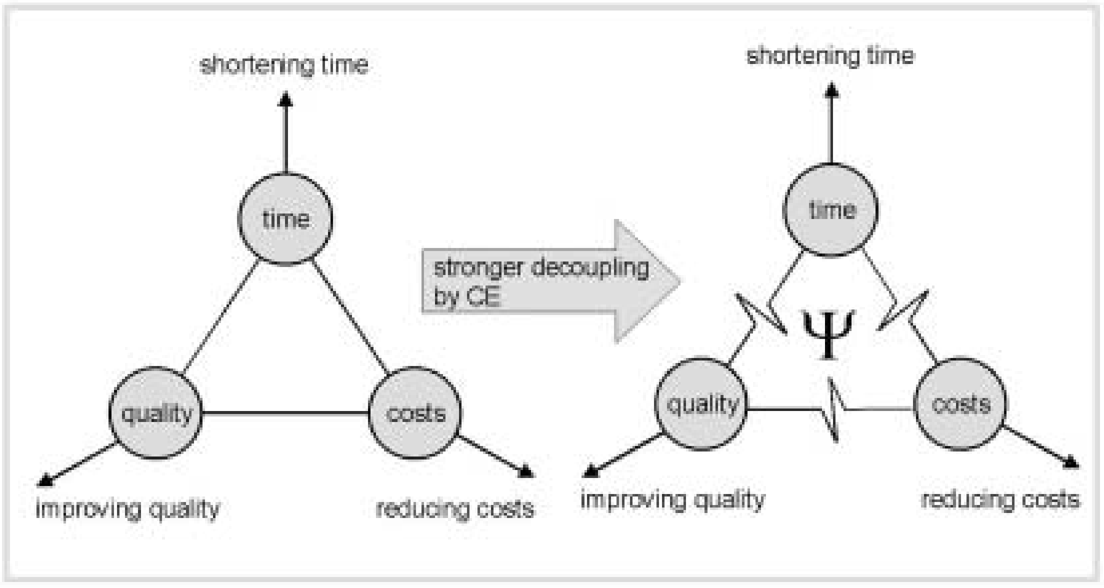


Figure 16: Decoupling of time, cost and quality by means of Concurrent Engineering (Berndes 1996)

- Cutting back barriers among departments and hierarchies.
- Promoting interdepartmental co-operation.
- Building up close links between suppliers and customers.
- Support of CE by top management.

### 3.3.3.1 Strategies for concurrent engineering

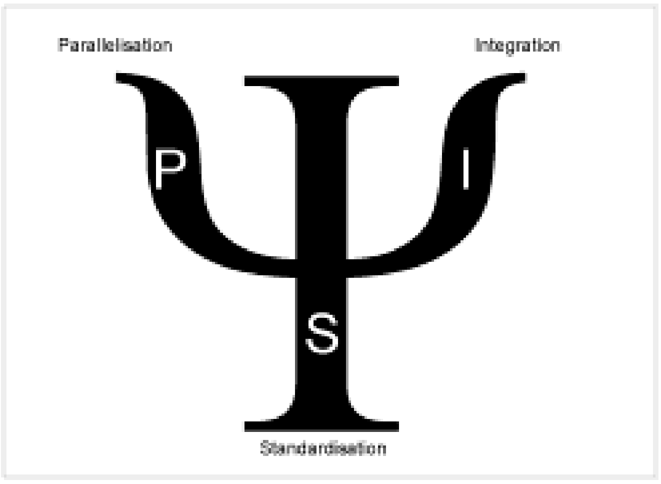


Figure 17: Strategies for concurrent engineering (PSI-strategy) (Berndes 1996)

Generally, three possible strategies can be identified as CE guiding principles (Figure 17):

- Parallelisation.
- Standardisation.
- Integration.

1. Parallelisation

Parallelisation in the product development process implies the cutting and optimisation of time. The first step is to remove existing float time in the development process. This means that processes, which do not have any dependencies on other processes, are carried out simultaneously. In practice most processes depend on others. In this case the dependent process has to be started before the preceding process is completed. An earlier start of the succeeding process is possible in most cases, because it can be carried out without having completed the preceding process. Not all information is required to start a new process. The result of this approach is the advantage of an accelerated execution of linked processes, but also the disadvantage of a higher decision complexity. This additional complexity is caused by an increased amount of information transfer between departments or teams. The proportion of uncertain and incomplete information is also higher due to the fact that not all parallel processes are finished which give inputs to other processes when they are started. In contrast to the Tayloristic principle, not only time can be cut due to parallelisation, but also amendment costs because of a lower number of mistakes. An example of parallelisation is the synchronisation of the development of product and means of production. The approach of parallelisation should be carried out under the principle that parallelisation does not mean to work side by side only but to work with one another.

2. Standardisation

Standardisation is defined as the unity of aspects in the product development process which show a high degree of similarity or the possibility of repetition. This is achieved by means of two basic approaches:

- *Structuring of processes*. Processes, which are often repeated, are specified and generalised.
- *Structuring of product*. This is the standardisation of products inclusive of its systems, elements and construction kit.

Standardisation is related to:

- Technical and structural aspects such as the usage of modules or components in the final product such as standard parts.
- Procedural aspects, structuring of operations and the definition of sequences of activities.
- Software standards such as ISO STEP ®, IAI IFC (Industry Foundation Classes).
- Aspects relating to the organisation of the structure such as interface between projects and departments. A clear definition of the organisation, i.e. standardised structures, is required to reduce and control the increased outlay of providing required information due to the implementation of CE.

The objectives of standardisation are to avoid repetition and needless work as well as to learn from existing experience of the company, industry or nation. Project staff can take repetitive and similar decisions quickly. Better co-ordination will be achieved. If routine tasks are optimised then theoretically more time will be available for innovative and creative work and for the management of unpredictable events. Standardisation should only be carried out if it is really necessary for parallelisation and integration (Berndes *et al*. 1996). Too much standardisation can lead to increased bureaucracy. Standardisation can vary from guidelines to compulsory arrangements and rules to fixed detailed operations.

3. Integration

If the product development process is seen as a uniform value-added chain then several departments such as R&D, sales, marketing, production and service are involved in the development of the product. The allocation of development tasks in different functional areas increases interface problems that result in the loss of information. The reason for the information loss is non-synchronised time scales, different interpretation of tasks and ignorance of the requirements at the other side of the interface.

Integration requires working in interdisciplinary teams and thinking and behaving in a process oriented way. There has to be the realisation that there is one common objective instead of several objectives that are department specific. The various departmental staff (or professionals in a construction team) must establish a view of the whole process that enables them to take appropriate action within their specific domains. Another important aspect of integration is data integration. A large proportion of construction data on large projects such as Menlyn is still in paper format. Integration will be greatly increased if more electronic information can be made reliable enough as to be trusted.

#### 3.3.3.2 Concurrent engineering enabling technologies

The successful implementation of CE requires a convenient-to-use information technology infrastructure. To achieve parallelisation, standardisation and integration and to introduce and support throughout the entire product life cycle process the CE platform consists of three major components (Kessler 1996:104):

- A framework to model, support, control and integrate processes and teams. All the data necessary for the product must be produced within these processes.
- An information Management System to manage, change, release and store metadata related to the product.
- A Products Information archive to store and access a common product data model.

Abovementioned requirements were identified in the ESPRIT project CONSENS (Concurrent Simultaneous Engineering System) in 1992. The objective of this project was to develop an organisational and information technology concept to realise Concurrent Simultaneous Engineering in European companies.

Typical features built into abovementioned to support parallelisation are:

- Controlled and concurrent access to distributed data and information.
- Multi project management.
- Client-server architecture and multiple desktops.
- Interactions between different software packages.
- Modelling of independent and dependent processes to enable parallel and simultaneous work.
- Flexible reaction to changes in the product development process as well as in the organisation of the project.
- Support the user in adapting the installed project according to new requirements.
- Distributed database.
- Common method for executing tasks via the user interface.
- Visualisation of information and data flows to and from other processes.
- Structuring of the project or product into distinct interrelated or independent work packages, which can be worked in parallel.
- Possibility to divide the project into its components and flexible management in work packages.
- To support teamwork and parallel access of team members to different tasks of an installed project.
- To provide mechanisms to free information or data in time for other users to enable concurrent and simultaneous work in this project.

Standardisation:

- Software is implemented on different hardware platforms in a heterogeneous network.
- Distributed databases.
- Standardised interfaces to exchange data between different software tools and frameworks.
- Support of standardisation with the possibility to reuse results in multiple projects and to ensure their consistency.
- To allow the reuse of results and work packages.
- To prepare libraries for the reuse of processes and project tasks.
- To support the installation and reuse of standardised processes and projects.
- To provide functionality to model processes and their data interdependence.
- Reuse of existing components by an interface connected to external documents handling systems and archives.

Integration:

- To integrate different kinds of users such as supplier, designer and project manager and to provide each user with his customised profile of the integration platform.
- To provide a common graphical user interface for each user of the system.
- To allow the integration of domain neutral tools to support the user in controlling the processes.
- To offer interfaces for communication and integration.
- Access to different tools through the user interface of the integration platform.
- To run the integration platform as a distributed system to support different user locations.
- To use standardised interfaces and mechanisms to allow the exchange of data between different design tools.

- Use of standards for communication and integration.
- To manage the status of results and keep track of their consistency so that completely specified information can be distinguished from partially specified information.
- To manage all interdependencies between work packages and to inform the participants of the effects of their task.
- To provide the possibility to exchange information in a controlled and defined way between different processes.
- To execute tools necessary to fulfil tasks in a convenient and controlled way.
- To check the data transfer between processes.
- Multiple schemas, interrelated data and a shared data model.
- Object-oriented structuring of the real world.
- Consistency control and storage in the data-handling component provides a defined status of the data.
- To support a project or process oriented organisation and changes in the organisation of the current project.

### 3.3.3.3 Flow management

The storing of architectural design knowledge in the form of cases or a smaller more portable format will be in the form of an encapsulated environment. This environment when it is brought into the specific design project will form part of a life cycle process. At this point it will be governed by the flow patterns of the specific process environment. For this reason the author is of the opinion that a study of the basic flows in the construction development process phases gives an idea of the information flows, but fails to clearly identify production capacity of the user types. This has the effect that the criticality of various decisions and the lead times required to ensure proper synchronisation in the fast track or concurrent engineering project cannot be planned for.

Activities used in a design process cannot be invoked in an arbitrary order as data and time dependencies of activities have to be taken account of. To enable the user the possibility to define a set of activities in a specific order, flows are introduced. A flow defines time and data interdependencies between activities. Specific features of CAD and CASE (in a software engineering sense) that are used in a design or software engineering process cannot be invoked in an arbitrary order. The output data of one activity might be required as input for another. Figure 18 illustrates the basic different types of flows that are possible in any process. The rectangular elements represent activities or processes. Their interrelations are symbolised by an arrow, which presupposes that the activity on the left of the arrow has to be executed before the one on the right.

Both in the ESPRIT SCENIC project and local studies undertaken by Allen at the CSIR in 1999 information flows between the different users of project information in construction were extensively studied. The SCENIC project identified the following generic stages in the building life cycle:

1. Inception.
2. Briefing.
3. Feasibility.
4. Concept design.
5. Scheme or outline design.
6. Detail design.
7. Tender documentation.
8. Estimating and tendering.
9. Evaluation of tenders.
10. Off-site fabrication or prefabrication.
11. Delivery or logistics.
12. Production or assembly.
13. Testing, commissioning and hand-over.
14. Operation and facilities management.
15. Re-use or demolition (disassembly).

The author noted that in all cases the links between the different users, the time of occurrence and the nature of communication were recorded. However there are two major omissions. *Firstly* the diagrams fail to identify the throughput capability of the various different constituent processes. No research has been done on the time taken to achieve certain design and decision taking activities. This analysis is critically important to implement a successful CE system within the construction industry. The *second* omission is the fact that certain activities or even processes should be grouped together to ensure efficiency and modularity. This has a direct influence on the future sustainability and maintainability of systems.

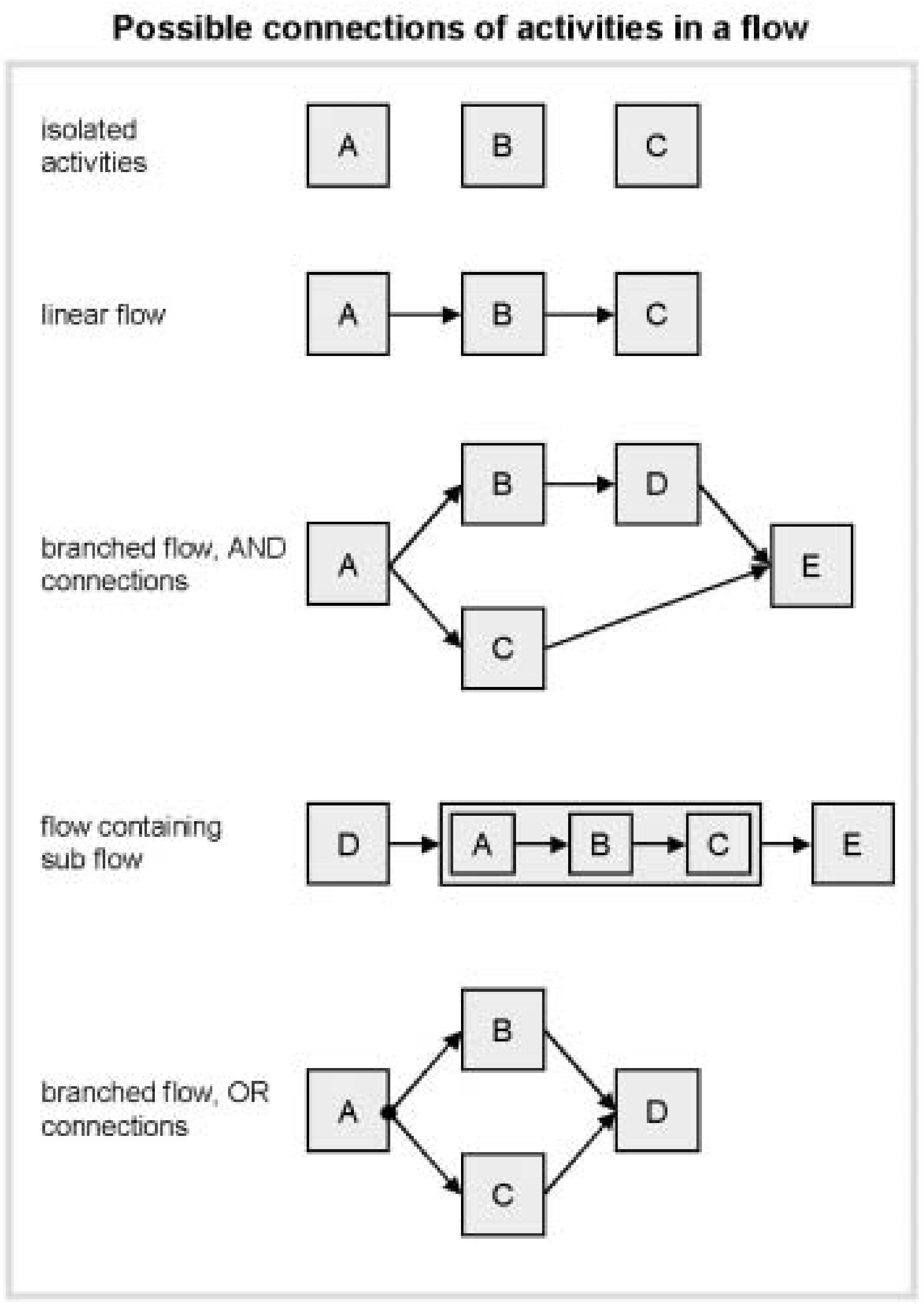

Possible connections of activities in a flow
isolated activities
A
B
C
linear flow
branched flow, AND connections
D
E
flow containing sub flow
branched flow, OR connections

Figure 18: Different flow types in a process (Author)

To the information in Figure 18 throughput information should be added (Figure 19). It is clear from a Voice of Customer (VOC) exercise, recently undertaken by the author that numerous bottlenecks can be identified during the construction process. Goldratt (1993:207) explained the impact on throughput in a process where bottleneck and non-bottleneck activities, equipment or even team members are combined (Figure 19).

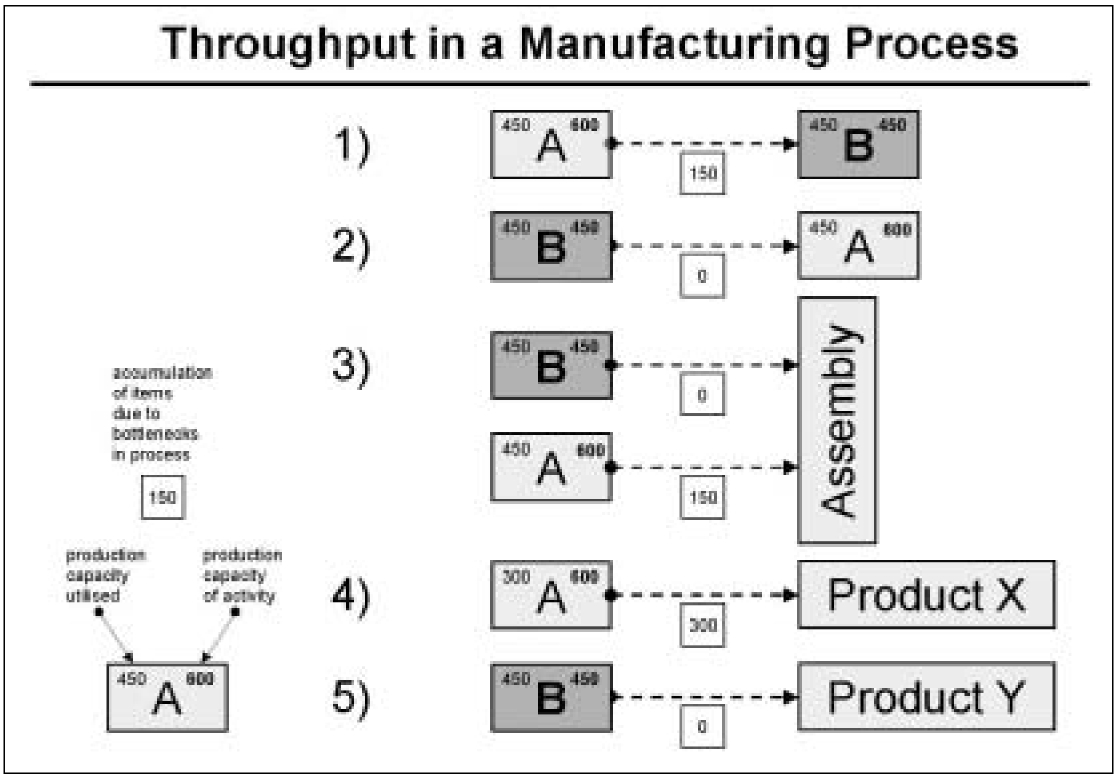

Throughput in a Manufacturing Process
1)
450 A 600
150
450 B 450
2)
450 B 450
0
450 A 600
3)
450 B 450
0
450 A 600
150
Assembly
accumulation of items due to bottlenecks in process
150
production capacity utilised
production capacity of activity
450 A 600
4)
300 A 600
300
Product X
5)
450 B 450
0
Product Y

Figure 19: Throughput in a manufacturing process (Author, based on Goldratt 1993:207)

In Figure 19, non-bottleneck activities (machines or workers) are designated with *A* and bottleneck ones with a *B*. The activities are connected by information or material flows. In each rectangular block the capability of the activity is indicated at the top right in bold numerals. The amount of the capability that can be utilised in each case is indicated by the value at the top left. The information flow consists of the following typical abstract and tangible activities and entities in the construction industry:

- Analyses (strategic, client and facilities analysis)
- Communication (appoints, reports, request, approves, program, brief, schedule)
- Design information (concept, scheme and detail design drawings, models)
- Construction material (pre-assembled and raw)
- Waste removal

Consider the various throughput types detailed in Figure 19.

Throughput type 1:

Non-bottleneck activity *A* is feeding bottleneck activity *B*. If *B* runs at full capacity, then 150 units will end up as inventory. In context this would mean unprocessed entities that cannot be handled by *B*. The production throughput of *A* is therefore effectively limited to 450 units.

Throughput type 2:

Bottleneck activity *B* is feeding non-bottleneck activity *A*. In this case, even if *B* runs at its full capacity, *A* will be starved of input, or it cannot run at full capacity. In this case only 450 units out of a possible 600 can be utilised. From this it can be concluded that the level of utilisation of a non-bottleneck is not determined by its own potential, but by some other constraint in the system.

Throughput type 3:

In this case output coming from both bottleneck and non-bottleneck processes are combined into a final product by means of assembly. In this simplified case, it is assumed that the final assembled product requires one item from *A* and one item from *B*. If process *A* runs at full capacity the effect will be that inventory (parts) will be manufactured that cannot be assembled into a final product, because too few parts are coming from the bottleneck *B* process. This has the effect that a significant amount of capital could be caught up in excess inventory. Goldratt (1993) identified the unexpected fact that a system running at full capacity is not necessarily an efficient system.

Throughput type 3 and 4:

In these cases there is no constraint in the system. Both *A* and *B* can produce at full capacity. However the constraint has now shifted from the internal processes to the ability to sell the products produced, in this case *X* and *Y*. Product X can only be sold at only 300 units per time unit. 300 units will therefore end up in inventory. In the case of product Y the sales force is able to sell all units per time unit. If the manufactured products cannot be sold, then capital will be tied up in inventory. If the sales tempo cannot be balanced with throughput the process could also become very inefficient.

#### 3.3.3.4 Theory of Constraints (TOC)

Theories are usually classified as either *descriptive* or *prescriptive*. *Descriptive* theories, such as the law of gravity, tell us why things happen, but they do not help us to do anything about them. Prescriptive theories both explain why and offer guidance on what to do. TOC is in essence a prescriptive theory. Goldratt states that several principles converge that make the manufacturing environment particularly applicable for TOC. Goldratt identified the following TOC principles:

- Systems thinking is preferable to analytical thinking in managing change and solving problems.
- An optimal system solution deteriorates over time as the system's environment changes. A process of ongoing improvement is required to update and maintain the effectiveness of a solution.
- If a system is performing as well as it can, not more than one of its component parts will be. If all parts are performing as well as they can, the system as a whole may not be optimal. The system optimum is not the sum of the local optima.
- Systems are analogous to chains. Each system has a weakest link (constraint) that ultimately limits the success of the entire system.
- The strengthening of any link in a chain other than the weakest link (constraint) does nothing to improve the strength of the whole chain.
- Knowing what to change requires a thorough understanding of the system's current reality, its goal and the magnitude and direction of the difference between the two.
- Most of the undesirable effects within a system are caused by a few core problems.
- Core problems are almost never superficially apparent. They manifest themselves through a number of undesirable effects (UDEs) linked by a network of cause and effect.
- Elimination of individual UDEs gives a false sense of security while ignoring the underlying core problem. Solutions that do this are likely to be short-lived. Solution of a core problem simultaneously eliminates all resulting UDEs.
- Core problems are usually perpetuated by a hidden or underlying conflict. Solution of core problems requires challenging the assumptions underlying the conflict and invalidating at least one.

- System constraints can be either physical or policy. Physical constraints are relatively easy to identify and simple to eliminate. Policy constraints are usually more difficult to identify and eliminate, but removing them normally results in a larger degree of system improvement than the elimination of a physical constraint.
- Inertia is the worst enemy of a process of ongoing improvement. Solutions tend to assume a mass of their own that resists further change.
- Ideas are not solutions.

Goldratt (1993) states that to be productive you must have accomplished something in terms of the goal. Productivity is meaningless unless you know what your goal is. If the goal is to make money, an action that moves the company towards making money is productive. The high level measurements that are normally used to measure company performance is:

- Net profit
- Return on investment (ROI)
- Cash flow

The primary goal of any company is therefore to make money by increasing net profit, while simultaneously increasing return on investment and simultaneously increasing cash flow. In practical terms this can be stated as in terms of the operational rules *throughput*, *inventory* and *operational expense*. *Throughput* (T) is the rate at which the system generates money through sales. It is specifically sales and not production, because if you produce something but do not sell it, it is not throughput. *Inventory* (I) is all the money that the system has invested in purchasing things, which it intends to sell. *Operational Expense* (OE) is all the money the systems spends in order to turn inventory into throughput. Everything that goes into a process is covered by the relationship between these three operational measurements.

In order to improve a system the question is where the attention should be focussed. The theoretical limit in reducing OE and I is zero. A system cannot produce output with no *Inventory* and no *Operating Expense* and they are therefore somewhat above zero. Theoretically there is no upper limit to how high you can increase T, but as is apparent from Figure 19, there is a practical limit to the size of your market. The potential for increasing T is likely to be much higher than the potential for decreasing OE and I. It makes sense to expend as much effort as possible on activities that tend to increase T primarily and make reduction of I and OE a secondary priority (Dettmer 1997:17).

### 3.3.4 Taguchi techniques for quality engineering

According to Ross (1988) Taguchi addresses quality in two main areas namely off-line and on-line quality control (QC). Both off these areas are very cost sensitive in the decisions that are made with respect to the activities in each. Off-line QC refers to the improvement of quality in the product and process development stages. On-line QC refers to the monitoring of current manufacturing processes to verify the quality levels produced. Off-line QC is of particular importance in this study due to the fact that it is proposed that CBR/ CBD methods will be used in the design process. It is also important to improve quality as early as possible in the product life cycle. Taguchi methods should be seen in context with the other important methods discussed such as QFD and Kansei engineering.

#### 3.3.4.1 The meaning of quality

Products have characteristics that describe their performance relative to customer requirements. Characteristics such as energy use of a house with regards heating of water, fuel economy of a vehicle and the strength of a door knob are all examples of products characteristics that are important to customers at one time or another. The quality of a product is measured in terms of those characteristics. Quality is related to the loss to society caused by a product during its life cycle. A high quality product will have minimal loss to society as it goes through this life cycle. The loss that a customer sustains can take many forms. It is generally a loss of product function or properties. Other losses are time, pollution and noise. If a product does not perform as expected the customer experience some loss. After a product is shipped, a decision point is reached. It is the point at which the producer can do nothing more to the product. Before shipment the producer can use expensive or inexpensive materials, use an expensive or inexpensive process, but once shipped, the commitment is made for a certain product expense during the remainder of its life cycle. This is of particular importance in the construction industry where the correct choice of lighting in a large shopping complex can save hundreds of thousands of Rands during the operational life of the complex.

Quality has but one true evaluator, the customer. The birth of a product is when a designer takes information from the customer to define what the customer wants, needs and expects from a particular product. Sometimes a new idea creates its own market, but once a competitor can duplicate the product, the technological advantage is lost.

### 3.3.4.2 Taguchi loss function

The Taguchi loss function recognises the customer's desire to have products that are more consistent and the producers desire to make a low-cost product. The loss to society is composed of the costs incurred in the production process as well as the costs encountered during use by the customer such as repair and lost business. A supplier in Japan made a polyethylene film with a nominal thickness of 0,991 mm that is used for greenhouse coverings (Figure 20). The customers want the film to be thick enough to resist wind damage but not too thick to prevent the transmission of light.  The producers want the film to be thinner to be able to produce more area of the material at the same cost. At the time the national specifications for film thickness stated that the film should be 0,991 mm ± 0,203 mm. A manufacturer that made this film could control film thickness to 0,02 mm consistently. The company made an economic decision to reduce the nominal thickness to 0,813 mm and with their ability to produce film within 0,02 mm of the nominal the product would meet the national specification. The intention of this was reduce manufacturing costs and increased profits.

Unfortunately at the time strong typhoon winds caused a large number of the greenhouses to be destroyed. The cost to replace the film had to be paid by the customer. These costs were much higher than expected. What the producer had not considered was the fact that the customer's cost would rise while the producer's cost was falling. The loss function, loss to society, is the upper curve. This is the sum of the producer and customer's curves. This curve shows the proper thickness for the film to minimise loss to society. This is where the nominal value of 0,991 mm is located.

It is clear from the function that as the film gets thicker from the nominal 0,991 mm the producer is loosing money. On the other hand when the film gets thinner the customer is loosing money. The producer should fabricate film with a nominal thickness of 0,991 mm and reduce variation to that thickness to a low amount. If the manufacturer does not attempt to hold the nominal thickness at 0,991 mm and causes additional loss to society, then it is worse than stealing from the customer. If someone steals R10-00, the net loss to society is zero. Someone has a R10-00 loss and the thief gained R10-00. If the manufacturer causes an additional loss to society, everyone in society has suffered some loss. A producer who saves less money than the customer spends on repairs has done something worse than stealing from the customer. Subsequent to this experience the national specification was changed to make the average thickness produced 0,991 mm. The tolerance was left unchanged at ± 0,02 mm.

The cost of damages to the environment through energy production (externalities) are also a loss to society.

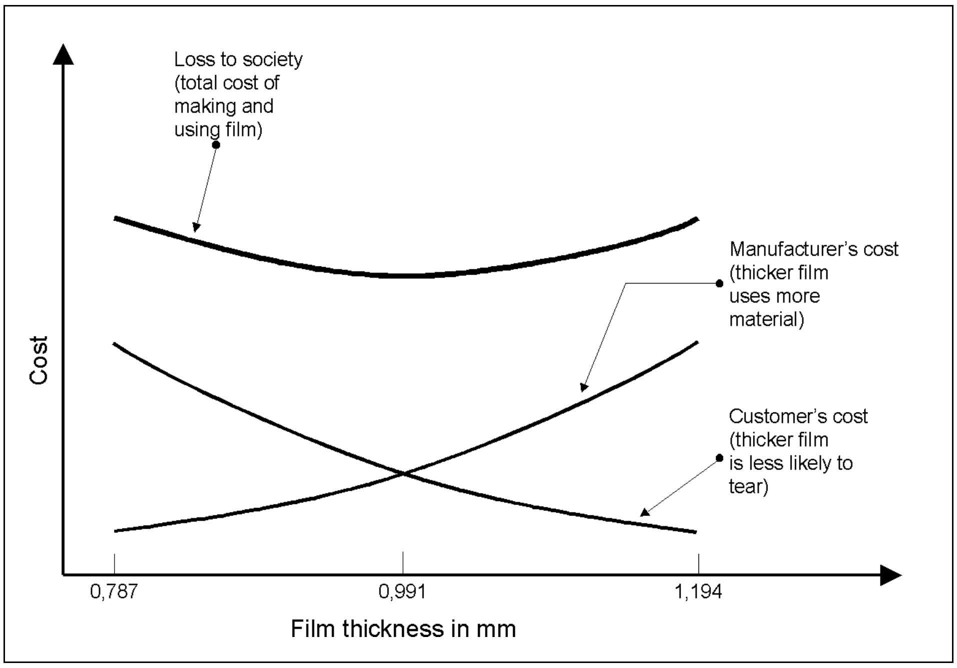


Figure 20: Costs associated with greenhouse film (Ross 1988)

## 3.3.5 The Fuzzy Front End (FFE)

In the high pressure environment of fast track (concurrent projects) time is an irreplaceable resource. The construction team should find opportunities to buy cycle time for less than cost. These opportunities appear throughout the development process. One place that is not often exploited seriously is the fuzzy zone between when a project opportunity is known and when we mount a serious effort on the development project. This approach is very different from conventional approaches that try to get a perfect solution at this stage by adding numerous checks and balances. The conventional logic is sound when markets are predictable and the cost of delay is low. However it breaks down in fast moving markets and when the cost of delay is high. This situation has been observed on large construction projects in this country.

Three critical factors combine to make the Fuzzy Front End an area of opportunity (Smith *et al.*, 1998):

- It lasts a long time.
- It is a cheap place to look for cycle time.
- Individual companies have big performance differences.

1. It lasts a long time.

Various delays occur right at the start of a construction project. There is a lot of time between when the team knows about a project and the time when a full development team started working on it.

2. It is a cheap place to look for cycle time.

If the typical actions that can be taken to buy a week of cycle time at various stages of the development process are analysed, enormous differences in cost are discovered. One consumer company spent $ 750 000 to buy three weeks of cycle time near the end of its development process by accelerating the shipment of critical capital equipment. This was a sound business decision, because the cost of delay on the programme was much higher than $ 250 000 per week. Yet the same three weeks could have been purchased for less than $100 a week during the FFE. This is 2 500 times cheaper!

3. Individual companies have big performance differences.

Some large companies plan so well that compelling market opportunities are lost due to the long period of time it takes to produce products. It has often been noticed that dynamic small companies can design and produce products long before the large company can even start. Instances have been noted where a small start-up company was 500 times faster than a large Fortune 500 company where well-intentioned planning and budgeting processes guaranteed defeat (Preston *et al*. 1998:52).

The following actions can be taken to improve the front-end processes:

- Institute metrics.
- Calculate the cost of delay.
- Assign responsibilities.
- Assign resources and deadlines.
- Capture opportunities frequently and early.
- Subdivide the planning.
- Create technology and marketing infrastructure.
- Create a strategy and a master plan.
- Prevent overloads.
- Create a quick-reaction plan.

Wheelwright *et al*. (1992:93) identified the primary types of development projects as:

- Enhancements, hybrids and derivatives.
- Next generation or platform.
- Radical breakthroughs.
- Research and advanced development.
- Alliance or partnered projects.

It is interesting to note the similarities between the construction industry and the *platform* development projects with regards the FFE. Platform projects represent the bundling and packaging of a set of improvements (design requirements) into a new system solution (design synthesis) for a much broader range of customer needs than the category of derivatives. Much creativity, insight and initiative are required at the FFE of a platform project than a derivative project.

## 3.4 Objects

### 3.4.1 Introduction

This section is included due to the dominance of objects in software engineering, CAD and interoperability. If the final application can be based on a generic platform using these technologies the likelihood of success is much higher. The author worked with one of the first object based CAD systems in the world i.e. GDS from Applied Research in Cambridge. The origins of this pioneering system through OXSYS, BDS and eventually GDS is described by Eastman (1999:53-61).

The concept of objects in software engineering, CAD and interoperability is dominating current software applications and new solutions proposed. Most modern programming languages claim to be based on objects. CAD systems such as MicroGDS, AutoCAD and MicroStation also claim to use object-oriented technology. To package software routines and data in the form of objects is very useful mainly because it keeps relevant, related data together. However objects are not the ultimate solution to all the problems studied in this thesis. This is mainly due to the fact that architectural design knowledge is generated at both tacit and explicit levels. The very fact that objects are encapsulated instances of a class implies that some higher order of system integration is required.  This chapter explores various theories surrounding this very broad subject. The chapter is concluded with an analysis of the main industry standard object technologies available. In order to be successful in the complex knowledge driven environment we are currently operating in, it is very important that objects conform to certain essential requirements such as interoperability, www enabling and platform independence.

### 3.4.2 Origins of the object approach

The central concept in the object approach is that of the object. An object associates data and processes in a single entity, leaving only the interface visible from the outside. The interface gives the user access to the operations that can be performed on the object. This approach is not new, it appeared in the language Simula. Simula was designed as a structured programming language for simulating parallel processes. The classes of Simula made abstraction possible by hiding the implementation and creating increasingly complex entities. The abstract aspect of the object approach, which enables a data structure to be hidden by the allowable operations for that structure was, formalised in the 1970s in the theory of abstract data types.

In parallel with this formalisation of abstract types, the language Smalltalk was developed. This language also implemented the object concept in the form of classes but added message passing taken from the actor concept and the use of inheritance to structure the classes hierarchically in terms of generalisation.

After Smalltalk the relations between generalisation and inheritance were developed extensively in artificial intelligence (AI) in connection with knowledge representation and more particularly in the context of frames and semantic networks.

Three points of view led to the object concept:

- Structural – the object is seen as an instance of a data type, characterised by a structure that is hidden by the permitted operations.
- Conceptual – the object corresponds to a concept of the real world, which can be specialised.
- Actor – the object is an active, autonomous entity that can respond to messages.

### 3.4.2 Why is the use of objects advisable

The object approach is characterised by the structuring of problems into object classes. The domains where this approach is used all require complex software that can handle large volumes of information. In the hope of controlling this complexity a number of objectives have been defined:

- Representation of real world entities without distorting or decomposing them.
- Re-use or extension of existing software.
- Development of environments rich in facilities such as tools for creating interfaces, debugging and for tracing execution paths.
- Rapid construction of high-quality graphical interactive man-machine interfaces, able to react to any external event such as a change of data.
- Facilitate the rapid prototyping of applications particularly for man-machine interfaces and general processing logic, without incurring the need for complete recoding.
- Facilities for exploiting parallelism when the software is implemented on multiple or distributed processor systems.

Objects can help the developer achieve these aims by their powers of abstraction, generalisation and interaction.

The object approach entails firstly defining the features of the objects that constitute an application and then making these objects interact by message passing. An object has a static aspect, which represents its state by means of instance variables or attributes. This is hidden by its dynamic aspect, which represents its behaviour and corresponds to the operations that can be performed on the object.

*Object-oriented design* is without doubt the main field in which the use of objects facilitates the work of the designer. It enables entities of the real world and the relationships between these to be represented directly. According to Meyer (1988) Object-oriented design is the method which leads to software architectures based on the objects every system or subsystem manipulates rather than the function it is meant to ensure. Object-oriented design is also the construction of software systems as structured collections of abstract data type implementations (Meyer 1988).

*Object-oriented programming* is unlike traditional programming. Objects are seen as active entities that perform their actions in response to messages sent to them. Instead of a software system program structure that consists of data and functions or procedures, a program is organised into active entities composed of data structures hidden by functions. The same function name can be used to perform similar actions on different objects, which makes it possible to construct an abstract language with which essentially different objects can be acted on in a similar manner.

An object approach program consists of a set of objects that exchange messages with each other, triggering operations (triggers or methods depending on the environment) that cause the internal state of the object to change and results to be returned.

An object-oriented database or object database (ODB) differs from traditional databases, because the real world objects are represented identically in the database on disk and in the application program in memory. An object is said to be persistent if its lifetime is greater than that of the program that created it and in this case it exists in the database. In traditional relational database management system an object is often decomposed in order to be stored into different database tables. In the object databases the object memory image is written directly to the disk.

It is estimated that by 1996 at least 80% of software developers were using object approaches (Bouzeghoub *et al*. 1997). One of the reasons for this gain in popularity is the fact that the object approach does not necessitate having pure object languages or pure object Database Management Systems (DBMS).

The market for object-oriented tools is very varied:

- Design methodology tools (CASE).
- Application Development Environments (ADE), including fourth-generation languages. These are usually used for building client-server applications for relational databases, using predefined graphical objects and a proprietary object language.
- Object-oriented programming languages (OOL) either pure object such as Smalltalk or hybrid such as C++.
- Object-oriented Database Management Systems (ODBMS). Either pure object or extensions of relational systems to include objects often called object-relational DBMS.
- Object-oriented middleware based on object request brokers (ORB) for passing messages between objects such as Common Object Request Broker Architecture (CORBA).

The four fields in which the object approach seems particularly important are:

- Programming, using either object-oriented extensions of existing languages (such as C, Pascal, Visual Basic) or pure object languages (Smalltalk, Eiffel or Java).
- Databases, using extensions to existing relational systems such as *Ingres*, *Oracle*, *DB2*, *Informix* or new systems, often based on OOL such as *Gemstone*, *ObjectStore*, *Versant* and *O2*.
- Design methods based on a combination of object-oriented models and specific representations. Objects have influenced most traditional methods.
- Distributed systems where distributed objects collaborate. As a result of the activities of the OMG, the object approach is bringing about a unification of the middleware products. This makes it possible to assemble objects over a network or even the Internet. Microsoft is important with its propriety approach at the core of Object Linking and Embedding (OLE) which is very likely to become a *de facto* standard in the near future.

## 3.4.3 Object-oriented programming

### 3.4.3.1 Encapsulation

Structured programming languages such as *Pascal* and *ALGOL* were designed with the aim of improving the structure of complex programs. They relate the processing to the data structure. In this approach there are three parts to an application program:

- Data structure
- Operations
- Main program

This approach was suitable if the application, albeit large, did not have to evolve a lot. It reaches it limits when the data structures or the procedures have to be shared by different programs and the data structures change with time.

Object programming solves these problems by encapsulation of the data and the operations that manipulate them in objects. This is an application of the principle of abstraction. An object is only accessible by means of its external interface operations that are visible. Its implementation is hidden from the programs that manipulate the object and have no effect on the programs that use it. Encapsulation thus ensures mutual independence among programs, operations and data. The advantage is that different programs can share the same objects without the need for import and export procedures.

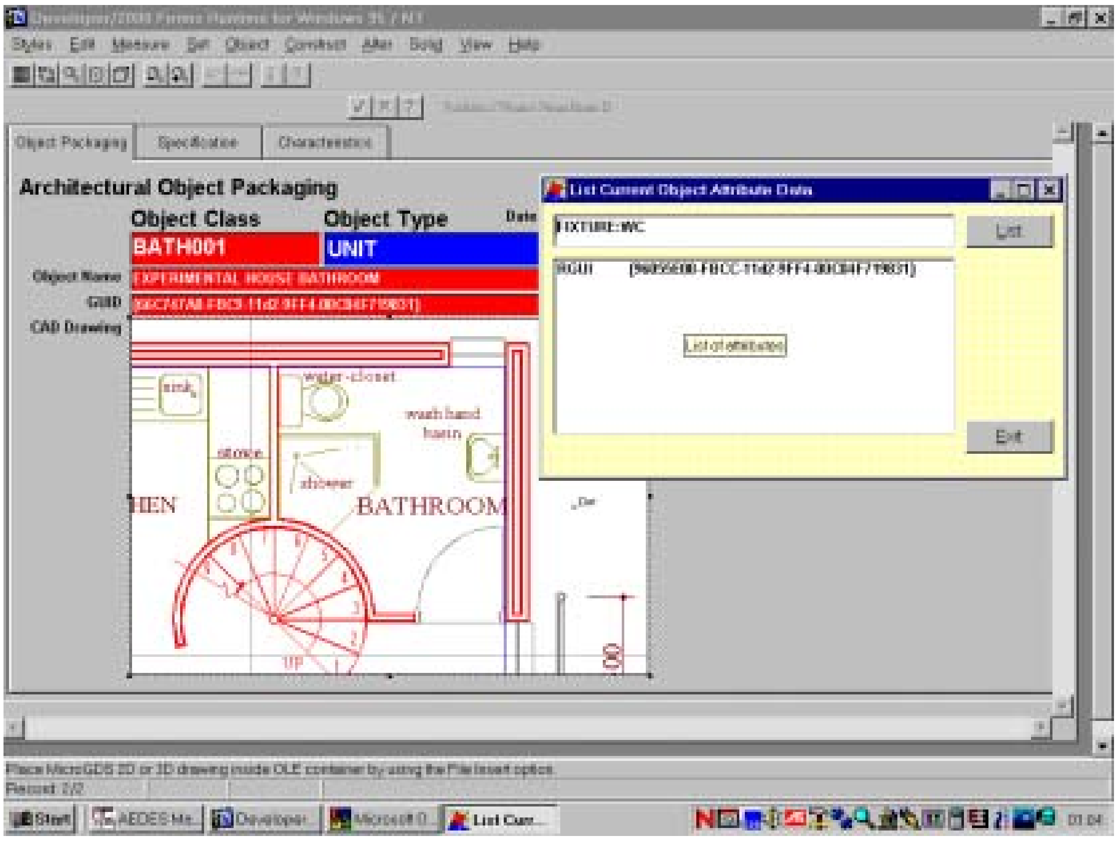
There are two very similar approaches that give a partial solution to the problems inherent in structured programming. The *modular approach* as used in the *Modula* language enables semantically related procedures to be grouped into modules that import or export procedures from and to other modules. This effectively encapsulates procedures in modules and thus makes it easier to represent the structure of an application. The *object-based approach* as used in the *Ada* language, extends the modular approach by adding abstract data types so as to make encapsulation of data structures possible. This increases the re-usability and extendibility of an application. Unlike the *object-oriented* approach the *object-based approach* lacks the concepts of inheritance and polymorphism.

#### 3.4.3.2 Objects

Object and class are interdependent. An object is an instance of a class. A class is a logical grouping of objects having the same structure and the same behaviour. An object is an abstraction of a data item and consists of:

**Object** = identity + behaviour + state

An *object identifier* (OID) defines an object's identity. It is an unique and invariant attribute that enables the object to be referenced independently of all other objects. In the *AEDES* prototype system the Microsoft Global Unique Identifier (GUID) was used to this effect for the unique and persistent identification of CAD objects. The identifier is either generated by the system where the object is created or is packaged with the object during construction. An example of the former is the implicit linking type object names used in *PREMIS*. An example of the latter is the GUID pioneered by Microsoft to ensure global unique identification of *ActiveX* controls. In *AEDES* this was used to ensure unique identification of the graphical packaging of the starter kits (Figure 21).



Architectural Object Packaging

| | |
|---|---|
| Object Class | BATH001 |
| Object Type | UNIT |
| Object Name | EXPERIMENTAL HOUSE BATHROOM |
| GUID | {66C747A0-FBC3-11d2-9FF4-00C04F719831} |
| CAD Drawing | |

List Current Object Attribute Data

FIXTURE:WC

RGUI {96055E00-FBCC-11d2-9FF4-00C04F719831}

Place MicroGDS 2D or 3D drawing inside OLE container by using the File Insert option.
Record: 2/2

Figure 21: Using a Global Unique Identifier (GUID) to link graphic objects to other data (Author)

Figure 21 illustrates various different object concepts. The CAD drawing that contains the layout of a small bathroom has been inserted into an Oracle form. The particular data field of the form is defined as an OLE container. By double clicking on the CAD drawing the server for the CAD drawing is activated. The server is actually the CAD program itself, but the user is not really aware of it. Once the CAD server for the particular drawing is running it modifies the menu structure of the Oracle form to reflect the capabilities of the particular application running. In this case the command language of the CAD package uses Visual Basic as its command language. This small autonomous Visual Basic application with the *List* and *Exit* buttons can retrieve the attributes of a particular object within the CAD drawing. In this case it is responding back with the RGUI of the FIXTURE:WC graphic object. The Global Unique Identifier (RGUI) has been generated by the Microsoft utility *guidgen.exe* that is distributed with the Visual Basic language system. The R in RGUI indicates that we are dealing with a particular instance of the class of objects called FIXTURE:WC. In the particular CAD system an R as the first letter indicates that the attribute data applies only to this particular reference or instance of the object.

The state of an object is a value that can either be simple, a literal, or structured, for example a list. In the latter case it can be composed of simple values, referenced to other objects or values that are themselves structured.

The behaviour of an object is defined by a set of operations that can be applied to it (the methods) and are defined in the class to which the object belongs.

An object is an abstraction of a data item characterised by a unique and invariant identifier, a class to which it belongs and a state represented by a simple or a structured value.

Two objects $O_1$ and $O_2$ are identical if their OIDs are equal. They are equal if their states are equal. For objects $O_1$, $O_2$ we write $O_1 == O_2$ if $O_1$ and $O_2$ are identical and $O_1 = O_2$ if their states are equal. This implies, $O_1 == O_2 \Rightarrow O_1 = O_2$.

**3.4.3.3 Class**

Objects of the same nature, for example a CAD representation of a building component, will generally have the same structure and behaviour. The class expresses the common features and is a means of classification.

**Class =** instantiation + attributes + operations

A class provides the mechanism of instantiation that enables a new object (design object in *AEDES* terms) to be created. The object that is thus created is an instance of the class. The set of all instances of a given class is the class extent. In the *AEDES* prototype eight hierarchical classes of architectural objects were identified:

| | |
|---|---|
| • Complex | (Hospital complex: Group of buildings) |
| • Facility | (Hospital building and site) |
| • Department | (Administration department) |
| • Unit | (Bathroom) |
| • Zone | (Wet area) |
| • Building Element | (Door) |
| • Component | (Door lock set) |
| • Sub-component | (Screw) |

A class is also an abstract data type, which specifies the attribute and behaviour of operations for object instances of the class. The instance attributes have a name and a type. The operations are the operations that can be applied to an object belonging to the specific class.

A class is an abstract data type characterised by a set of properties (attributes and operations) common to its objects, with a means for creating objects with these properties.

The principle of encapsulation ensures that the attributes of a class can only be accessed externally by means of the operation (method) of reading that is provided.

## 3.4.4 The model approach to architectural design

In the past many attempts were made to structure architectural design in the form of full 3D models. One of the earliest attempts was *OXSYS*. All these attempts failed to achieve full automation of the design process or full quantification of the various design factors. Richens (1994) states that the Knowledge Based System (KBS) community is over-optimistic in their analysis of the nature of design. The KBS community (Carrara *et al.* 1994) typically characterises design as:

- Defining a set of functional objectives that ought to be achieved by the design artefact.
- Constructing design solutions which, in the opinion of the designer, are (or should) be capable of achieving the predetermined objectives.
- Verifying that these solutions are internally consistent and achieve the objectives.

However, the abovementioned is only a part of what really goes on in architectural design. Architectural objectives usually include functional ones, but are dominated by less definable intentions and are never collected completely before design starts. They evolve and are discovered as the work proceeds.

This is not due to architects that are badly trained, but more due the nature of the problem. Yet an approach that starts with a requirement that leads to derived functions and to design objects is useful if the intention is to package the explicit design knowledge. Pugh (1996) states that design is not only the integrative mechanism that brings together the arts and sciences but also the culture which envelopes both. Design is not like mathematics or physics. It does not represent a body of knowledge. It is the activity that integrates the bodies of knowledge present in the arts and sciences. The author is of the opinion that design is both an integration between arts and sciences in a horizontal dimension, but also an integration between tacit and explicit knowledge in a vertical dimension over time.

The reasons that could be identified why an electronic model approach to architecture is only partially successful and why attempts towards interoperability are likely to fail are the following:

- Model based approaches assume that all design knowledge must reside in a single model, or a set of interoperable objects. This approach assumes that all design knowledge can be quantified.
- It is assumed that working drawings such as plans, elevations, sections and details can be produced from the model. In *OXSYS*, *BDS* and other subsequent systems this failed because architectural drawings contain notations such as specifications, dimensions and general notation that can only be conveniently placed on 2D extractions of the 3D model. These extractions to 2D worked well, however if any changes are required then changes had to be made in the 3D model. This was time consuming and inefficient.
- The model approach captures only the functional manifestations of the design process, not the invisible tacit and experience aspects.
- Intelligence in a CAD system is inversely related to flexibility. The more detailed you wish to make the data model, the more circumscribed is the Universe and so flexibility is lost.
- Even after 5 years the International Alliance for Interoperability (IAI) failed to solve the problem of the intelligent internal interoperability of design objects such as a door in a wall. The IAI is likely to achieve extraneous interoperability due to standardisation of the design objects. Even at this stage the definitions of internal design objects are very incomplete and still require a significant effort to bring about a new industry standard.

### 3.4.5 Frameworks for object components

The use of objects improves technologies such as languages, tools, databases and other technologies that are essential for the construction of complex applications. The objects that can be produced by means of these technologies are very heterogeneous, particularly in size and level of abstraction. To assemble these into an application creates very difficult problems. The task can be simplified by using middleware products such as *CORBA* for communicating between these heterogeneous objects.

An *object component* is an independent, autonomous object, capable of co-operating with other objects in a distributed system with the intention to provide a globally available service to the application. Components were initially developed to ease the work of system developers and integrators where the synonymous terms *technical object* and *technical component* had their origins. More recently the concept of *business object* came into use.

Object middleware is a software bus that enables any object component to inter communicate in a manner that is transparent to the network and to the specific software application. It must be convenient to create, deploy and maintain complex systems. The components should be extendable and adaptable to specific needs and be capable of being combined dynamically in many different ways. Object frameworks offer an architecture into which the object components can be integrated and co-operate. An object framework provides tools for creating and assembling the components. The development of such frameworks, a very important requirement in the object world, is a great challenge. Only a few designers have mastered the technology of middleware. Recently frameworks for compound documents such as Object Linking and Embedding (OLE) developed by Microsoft appeared. The metaphor of a document is used for the integration of components.

### 3.4.5.1 Aims of object components

A component should have the following properties:

- Standard interface.
- Encapsulation
- Extendibility.

Standardisation of interfaces is the only way to ensure that independent components, developed with different programming languages, can co-operate. Encapsulation ensures that as components evolve they will not impact on the interface and the application that must use it.

The creation of abstract superclasses improves the extendibility of a software product. The superclass defines the general behaviour common to all possible classes. If a new special subclass is desired all you have to do is to implement a new subclass with only the specialised behaviour that is different.

The term object component includes many different software products such as libraries of classes, graphical interfaces, compound-document components and business objects. Three main categories can be identified:

- Technical components.
- Compound documents.
- Business components.

### 3.4.5.2 Technical components

These are libraries of classes developed by programming languages such as *C++* or *Smalltalk*. They are normally generic and extend the services provided by the language. In the *PREMIS* parametric symbol programming language, *Symbolix*, special components are provided to facilitate the programming of complex graphic visualisations. They are sometimes specialised for a particular activity such as access to relational databases or the development of graphical interfaces. The use of technical components is more along the classical lines in an *object-oriented* manner, by including classes from these components in the application program.

### 3.4.5.3 Compound documents

A compound document is an electronic document into which diverse components can be incorporated. Such a document is typically presented as a graphical window in which each component has its own graphical interface. The document metaphor is used extensively in the personal computer world. The document is generalised to enable objects of a variety of types to be edited and visualised in a uniform manner. In contrast to the cut and paste approach to joining heterogeneous objects, the compound document approach is essentially dynamic. This enables the object component to be manipulated directly from the document in which it is contained. This has the benefit of simplifying the creation and maintenance of complex documents.

Use of this approach requires many document components to be available in particular multimedia components. Technical components are easily integrated into a compound document.

#### 3.4.5.4 Business components

A business object is an element of a typical business field of activity such as that of a bank or manufacturing company with rich semantic properties such as name, attributes, interfaces, relationships and constraints. Typical business objects for a manufacturing company may be address book, customer management, warehouse management, cash management, purchases, orders, finances, parts and deliveries. The objects must be able to communicate at a high semantic level. The company Discon uses a business object approach (Engelbrecht 1998). They use the technique of Functional Effect Back tracking to calculate the most appropriate grouping of these objects. The intention is that if a major component such as the financial module is replaced with a newer one it should not in any way destabilise the existing rest of the components.

Business components are of large size, with classes composed of many subclasses. They are unlike technical components and components of documents that are designed to facilitate the re-use of code. They are created in a top-down manner by means of object-oriented analysis and design.

Due to the availability of powerful CASE and 4GL tools more and more time is devoted to the analysis and design of business systems. Business components are becoming increasingly important in giving structure to applications. Enterprises are developing these components to meet their own needs for interoperability and re-use of applications.

The Business Object Model Special Interest Group (BOMSIG) of the OMG is working on the standardisation of business objects. It proposes to define these in terms of three types of object:

- Presentation, for managing the object’s graphical interface.
- Business, for managing the object on disk.
- Process, for realising complex operations on the business objects.

Objects of these three types can be heterogeneous and will communicate over a distributed architecture such as *CORBA*.

### 3.4.6 OLE/ COM from Microsoft

In 1994 Microsoft began to introduce Visual Basic custom controls. Today these controls are known as *ActiveX*. Microsoft is actively promoting the use of a new object-component model (COM) for the efficient management of distributed objects and the incorporation of these into compound documents. Today it is possible to use these controls in a wide variety of environments such as *Microsoft Word*, *Excel*, *Visual Basic*, *Access*, the www and other third party products such as *Arena* and *Visio*.

The problem of the fragile base class in classical models such as *C++* inhibits distribution of components and particularly successive versions in binary. Classical object models are therefore unsuitable for distribution over a network. This problem motivated the development of dynamic languages such as *Java*. Microsoft's aim of code exchange in binary implies the need to define a means for invoking objects at binary level, independent of any language. The *CORBA* approach, which specifies the interface in terms of a language, is not suitable.

Management of compound documents is an important objective of producers of distributed component frameworks. Microsoft is seriously committed to this route with its OLE architecture that is based on Component Object Model (COM).

There are two possible methods for integrating objects into a compound document, i.e. linking and embedding. The linking option uses a pointer in the forms of a name or identifier into the document. The linked object continues to reside in the original source document. The advantage of this is that it does not increase the size of the document and allows for multiple applications to share the object. In the case of embedding a copy of the original source object is inserted into the current document. This increases the size of the document but allows the linked object to evolve independently.

A framework for compound documents offers various basic services when components are assembled, stored and modified. The main services are persistence, saving objects, data exchange, construction of documents and interactive activation for making these active and modifying them.

#### 3.4.6.1 Persistence of objects

Containers can be saved to disk and returned to memory when needed. It is important to remember the type of every object component. Microsoft organises the objects into compound files each containing a number of subfiles.

#### 3.4.6.2 Data exchange

The facility of exchanging data between documents enables parts of documents to be copied directly into another. It is an essential requirement for drag-and-drop. This service can make use of the more rudimentary cut-and-paste service that uses the clipboard as an intermediary place of storage.

#### 3.4.6.3 Enabling relationships between documents

A basic relationship service is necessary for enabling one document to point to another or to an object. In the case of simple compound documents the relationship is logical inclusion. The management of hyper documents requires other types of links to construct the hyperlinks. For compound documents aggregation is the only type of link that is handled. The invariance of the pointers creates special problems. Use of absolute file names with offsets inside the file is prohibited so as to ensure this invariance in the face of modifications. OLE uses names of files or subfiles, but these are modified if the reference is changed.

#### 3.4.6.4 In-place activation

Activation occurs when the object is selected in the document window. The document remains under the control of its application or of an adapted application. This facilitates visualisation and editing. The editing menu of the main application is changed to reflect the needs of the object under consideration whilst maintaining full synchronisation.

In the case of OLE this is achieved by contacting the application that created the object under consideration. It continues to load it under a server process (if it is not already active) and transfers control to the server.

### 3.4.6.5 The object-component model

COM supports classes. A class is being implemented as a set of functions provided by servers in the form of executables (EXE) or by libraries (DLL). Encapsulation is total and data are hidden. Only the functions are visible. A class has a 32-bit global identifier (GUID) that is generated by an OLE utility or for basic classes specified by Microsoft.

An object is an instance of a class, with hidden data. In general it will have several interfaces each with an identifier and accessible at binary level by a pointer. Clients communicate with an interface by means of this pointer. This can be obtained through the medium of the interface identifier. It references a table of functions of the interface, given in the order they were specified.

To enable users to find the interfaces, each object is provided with a standard interface *IUnknown* through which its other interfaces can be found. The interface to *IUnknown* is thus a root from which all classes providing basic functions descend. It enables a user to point to any object and obtain at least a pointer to *IUnknown. IUnknown* also enables counts of references to be kept, so that the memory space of objects no longer being referenced can be reclaimed.

Inheritance of structure does not exist in COM. Inheritance of structure is replaced by objects with multiple interfaces. This enables modification of components without incurring the need for recompilation. It also makes versions of interfaces possible. The existence of one interface through which the others can be found means that objects are self-documenting.

### 3.4.6.6 Support for distributed objects

This enables access to the interface of distant objects, managed by a different process EXE. The server process can be local or distant. OLE ensures transparency of the type of server for all client objects. Different components are brought into play for this purpose and a service is provided for localising the server and initiating execution. This service, service control manager (SCM), localises the server by first consulting the system directory and then activating the server. If it is already active it will establish contact with it. Message passing is based on the RPC of DCE. It is a matter of creating an issuing proxy object in the client and a receiving stub in the server. This mode of dialogue takes place over a proprietary software bus.

### 3.4.6.7 OLE/ COM basic services

COM and OLE provide the services essential to distributed systems. The basic services are implemented more in COM and those relating to compound documents more on OLE. Some of the services provided are:

- Persistence, for storing and retrieving distributed persistent objects.
- Exchange, for transferring data between components in a uniform manner.
- Relationships, for implementing links by means of intelligent names.
- In-place activation, for assembling and activating multiple components in a container.
- Automation for dynamic invocation of typed objects from programming languages such as Visual Basic and Visual C++.

The functions essential to these services are described below. They are grouped according to the interfaces to the objects that use the services.

*Persistence*

This provides for saving and restoring collections of objects either directly or by means of links within a single file. The container receives the application objects in memory pages and its contents can be saved in a compound file. There is therefore a hierarchy of files consisting of data elements and directories held in a single physical file.

Transaction management is used. Writing is in transaction mode, with either complete validation at the end of the transaction (*commit*) or cancelling all the updates performed in the course of the transaction (*revert*). Files can be modified incrementally without the need for a complete rewrite. Files are shared, making it possible for data to be exchanged between processes. The basic interfaces provided by the file management service are as follows:

*IStorage(Record->Create,Open,Copy,EnumElementTo, …)*
*IStream(File->Create,Read,Write,Seek, …)*

*Data exchange*

COM provides a uniform data transfer service, for which there is a standard interface *IDataObject.* This interface provides functions for recording in memory and retrieving data in various formats. The basic functions are *GetData*, *SetData* and *EnumerateFormat.* Objects with the interface *IDataObject* can exchange data directly without needing to go through a clipboard as is required in Windows. The interface also enables a user to be advised of a change in the source.

On top of *IDataObject* the drag-and-drop service enables data to be moved from a source object to a target. This is by means of a pointer to an *IDataObject.* The service acts as a mediator between the source and the target. The two must have the interfaces *IDropSource* and *IDropTarget* respectively.

*Relationship*

This enables objects to be linked to or incorporated into compound documents. It manages relationships that are simple aggregations. A link references an object from a compound document. It is implemented by a persistent name, a *moniker*. A moniker is an object that implements a link. Monikers support composite names, relative names and a function *BindToObject* that gives access to an object.

*In-place activation*

An object that forms part of a compound document is activated by a double click. The application that created the document is loaded as a server and takes over control. It can act either in-place or in a new window. The document can be edited directly on-screen. Linked or incorporated objects can also be activated in place, without the need to create a new window. The application takes over the document window and the object becomes the active agent that controls the keyboard. Only incorporated objects can be modified. Several interfaces are needed for in-place editing. The container must support *IOleInPlaceSite* and *IOleInPlaceFrame* and the object must have been given *IOleInPlaceActiveObject.*

Automation is a key OLE service that enables the functions of an application to be described (EXE), incorporated into OLE and made dynamically callable by any language that can use scripts. A good example of this is the *VBScript* sub-set of Visual Basic that is now extensively used in Internet web pages. The interface is described in the *Object Description Language* (ODL). Object descriptions are held in a *type library*. The type libraries can be managed directly by the *ICreateType* interface for creation.

A client OLE automation controller invokes an OLE automation server. The invocation is dynamic and is passed by the interface *IDispatch*. *IDispatch* receives the function Invoke from the client, decodes its parameters and sets up a link with the server. It calls the required function and passes the parameters.

### 3.4.6.8 The main OLE interfaces

Microsoft introduced the OLE architecture in 1991 for the purpose of managing compound documents. Since 1994 all the interfaces described above have been combined in a single context. Multiprocesses are supported in Windows NT and single processes in Windows 95/ 98. Recent distributions of Visual Basic include the distributed version of COM, DCOM. This has the effect that OLE is steadily evolving from its original use in compound documents to a distributed system where any client can communicate with any server.

The kernel implements COM and brings together the basic functions of persistence of objects, management of intelligent names and uniform transfer of data between objects. OLE is built on top of COM or DCOM. It provides management of compound documents by means of in-place visual editing and drag-and-drop of objects between documents. It also supports nesting of linked or copied objects and management of relative links.

Microsoft provides a very complete architecture. It is object based rather than object-oriented. Unfortunately it does not support inheritance. The services and framework provided is very complex. The architecture goes well beyond the handling of compound documents. Distribution is achieved by means of message exchange between proxies and stubs using RPC.

Objects are multi-interfaced providing a kind of multiple inheritance with convenient properties. Microsoft is already using a component approach, based on OLE in many fields such as operating systems, databases and multimedia. Other suppliers can enhance Microsoft’s products by incorporating their own components.

## 3.5 Kansei engineering and new product development

### 3.5.1 Introduction

Kansei Engineering (KE) is one the lesser-known product development techniques. Due to the enormous influence that the Japanese had in the domain of product development this technique was included. Figure 2 indicates that KE is one the techniques that is able to extract tacit needs from the level of *unexpressed thought*. This is exactly the area in design where very little has been achieved in design systems over the past 35 years. KE is also one of the few proven techniques that can operate at this high tacit level. It is envisaged that some support for KE be included in the final product.

The term KE was first used in 1986 by Kenichi Yamamoto, the current chairman of Mazda Motors, in a special lecture given at the University of Michigan. Thirty years ago companies could easily make a profit because it was a seller's market. The product development strategies of the time were based on product output. With the increasing saturation of the market product developers and marketing had to pay increasingly more attention to quality to differentiate their products. Today consumers demand good quality products. The economic success of a manufacturing firm depends on their ability to identify the needs of customers and to quickly create products that meet these needs and can be produced at low cost (Ulrich *et al.* 1995:2). To achieve these goals is a marketing, design and manufacturing problem. The totality can be called product development or total design (Pugh 1996). The basic needs manifested itself in movements such as the Total Quality Movement (TQM). Moss (1995:4) states that TQM is both a philosophy and a set of guiding principles that represent the foundation of a continuously improving organisation. TQM is the application of quantitative and human resources to improve the material and service supplied to an organisation. It is also the degree to which the needs of the customer are met, now and in the future. KE is one of the methods that can be used to quantify the higher order tacit feelings of the consumer into a product.

Nagamachi (1999a) is of the opinion that even in a strong economy, like that of the U.S.A., one of the reasons why some products sell well and some not is due to fact that not all products focus on consumer feelings and emotions. He calls this Kansei. The use of KE could provide quantified knowledge that can potentially be encapsulated in object-oriented architectural design packaging (Figure 1). Zultner (1999:360) identified KE as one of the methods to define customer needs that concentrate on the emotional responses.

### 3.5.2 What is Kansei Engineering (KE)

It is a technology that attempts to quantify cognition and product image in such a way as to influence the product development process (Figure 22). Like Quality Function Deployment (QFD), Kansei is a technique that is consumer-oriented and attaches importance to the voice of the customer. Nagamachi started KE at Hiroshima University about 25 years ago. It is an ergonomic consumer-oriented technology for new product development. In Kansei engineering Nagamachi focuses on three main aspects:

- Accurately understanding of consumer Kansei.
- To translate the quantified Kansei values into the product design.
- To create a system and organisation for Kansei-oriented design.

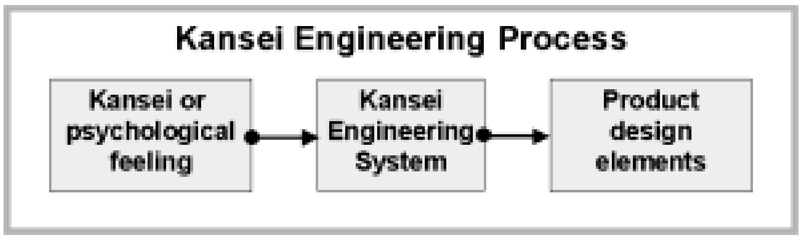

Kansei Engineering Process
Kansei or psychological feeling
Kansei Engineering System
Product design elements

Figure 22: The Kansei engineering process (Nagamachi 1999)

The word Kansei encompasses the following meanings:

- A feeling that one holds about a certain thing that may or may not exist but is thought to help enhance one's quality of life.
- All feelings and emotions that one has about a product, including its functions and appearance.
- Vague psychological emotions and senses that one holds but is not yet expressed.

The techniques used in Kansei attempts to quantify human *cognition* through the six senses of *vision, hearing, smell, taste, touch* and *inner sense (feeling).* For example you would walk into a building and approach the receptionist's desk. You rapidly form a first impression. You might feel that interior design is very modern *(vision),* the receptionist is very professional *(cognition)* and the acoustics is very good due to the soft carpets, wall and ceiling construction *(hearing).* Kansei initially sounds vague and ambiguous. However with special methods the feelings can be made tangible for the specific product where they manifest.

KE is a technology to translate Kansei into the design domain. Presently well-developed methodologies and software systems support it. The general input sources of KE data could be:

- Physiological data. When using physiological data to measure Kansei, the software can be used to determine data patterns.
- Psychological data can be handled by means of the sorting of data into clusters using a neural network model, classification of the data by means of genetic algorithms and breaking down the data into design elements using the quantification theory.

The latter method is most often used because of its convenience.

### 3.5.3 Types of Kansei Engineering

Five main technical types of KE is most often used:

- Type 1: Category Classification. It identifies the design elements of the product to be developed, translated from the consumer's feelings and image.
- Type 2: Kansei Engineering System (KES). A computer aided system, with an inference engine and Kansei databases, is used.
- Type 3: Hybrid Kansei Engineering System. The dual software systems of forward KES that goes from Kansei to the design specifications and reverse KES that goes from design specifications to Kansei is used in this type.
- Type 4: Virtual Kansei Engineering. This is an integration of virtual reality technology and Kansei engineering in a computer system.
- Type 5: Collaborative Kansei Engineering Designing. Group work design system utilising intelligent software and databases over the Internet.

### 3.5.3.1 Type 1: Category Classification

Category Classification is a method by which a Kansei category of a planned target is broken down into a tree structure to determine the physical design detail. Mazda used this type of KE for the new "Miata" (Eunos Roadster in Japan). KE became the fundamental technology for new product development at Mazda.

In the case of the "Miata" the project team decided that the zero level product purpose (mission or aim) would be "Human-Machine Unity" (Figure 23). The team broke the zero-level concepts into subconcepts level 1, 2 to n. In the case of the "Miata" the zero-level concepts were *tight feeling*, *direct feeling*, *speedy feeling* and *communication*. The various feelings are translated into physical traits, ergonomic specifications and automotive design elements.

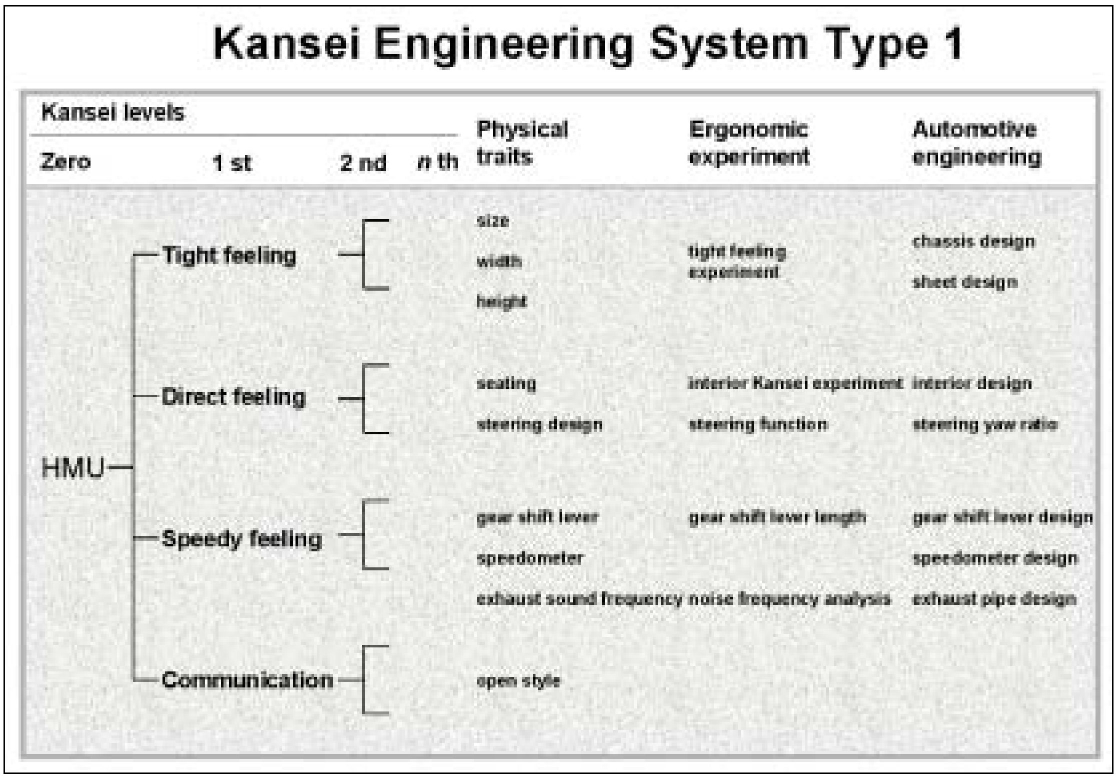


Figure 23: The translation of Kansei into physical car traits (Nagamachi 1999)

### 3.5.3.2 Type 2: Kansei Engineering Computer System (KES)

The KES is a computerised system with an Expert System that supports the transfer of the consumer's feeling into physical design elements. The KES has four databases and an inference engine in the KES structure (Figure 24). The following databases are used:

- Kansei Database. Kansei words used in the new product domain are collected. Typically 600 to 800 words are initially collected. This is reduced to approximately 100 words that best describe the new product. These words are normally adjectives and sometimes nouns. In the automotive industry words such as "fast", "easy to control" and "gorgeous" are used. After an ergonomic evaluation has been conducted these Kansei words are analysed by multivariate techniques such as factor and cluster analysis. The Kansei database contains the statistically analysed data. The KE is conducted by means of a Semantic Differential (SD) method on a five-point scale.

- Image Database. Data evaluated by SD scales are then further analysed using Hayashi's Quantification Theory Type II (Nagamachi 1999b). It is a multiple regression technique for qualitative data. The statistical relationships obtained between Kansei words and design elements constitute the image database. This database relates the most appropriate design elements to Kansei words and vice versa.

- Knowledge base. The knowledge base is a rule based database in *if then* form. It controls the image database. It also includes design guidelines and digital colour expression system.

- Shape and colour databases. The design detail is implemented in a shape design and colouring database. The parts are design aspects that are co-ordinated into the final assembled product with each Kansei word. The colour database consists of colours co-ordinated with Kansei words. The design and colour is extracted by a purpose made inference system based on the rule-base and is graphically displayed on the screen.

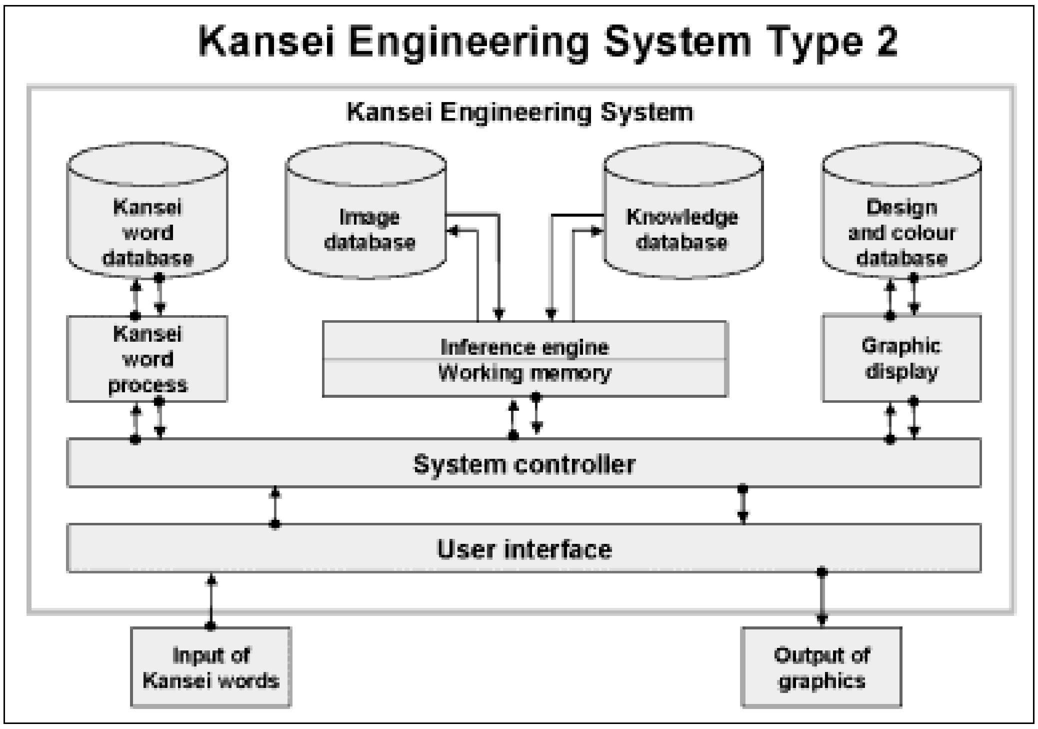


Figure 24: Type 2: Kansei Engineering Computer System (KES) (Nagamachi 1999)

#### 3.5.3.3 Type 3: Kansei Engineering Modelling

KE type 3 uses a mathematical model constructed in the computerised system in stead of a rule-base system as described above at type 2. The mathematical model is based on Fuzzy Logic. Sanyo attempted to use Fuzzy Logic in an intelligent colour printer that could enhance bad original images. The developers carried out an experiment on an image of a beautiful girl and obtained data of the hue, brightness and saturation for a girl's face colour which are represented by a membership function in Fuzzy Set Theory. The data were transformed to Red: Green: Blue in the computer colour system. This enabled the KE colour printer to enhance the original picture by means of the KE inference system. The intelligent colour printing system comprised a camera, computer and a colour printing system driven by Fuzzy Logic. It was able to diagnose the original picture.

Nagamachi also developed a computerised language analysis system for the Japanese language to analyse words in terms of Fuzzy Integral and Fuzzy Measure Logic. It is used to analyse brand name feeling. Several Japanese companies use this system to select appropriate product names (ring and feeling) for new brand products.

### 3.5.3.4 Type 4: Hybrid Kansei Engineering

In contrast to the forward KE discussed under type 2, type 4 is called Backward KE. This is used in the situation where there is an existing product and the designer wants to know how well this design fits a specific set of Kansei criteria. The computerised system makes design suggestions. If types 2 and 4 are combined then it is called a hybrid KE. (Figure 25).

By means of this type of KE system the designer is able to get design specifications from Kansei words through the forward KE. This helps the designer to be more creative based on his own ideas and suggestions offered by the system. The drawings generated can be input into the system. An image processing system can analyse the sketch. The system is then able to diagnose the input sketch by reference to the Kansei database. This enables the designer to evaluate his own creative design.

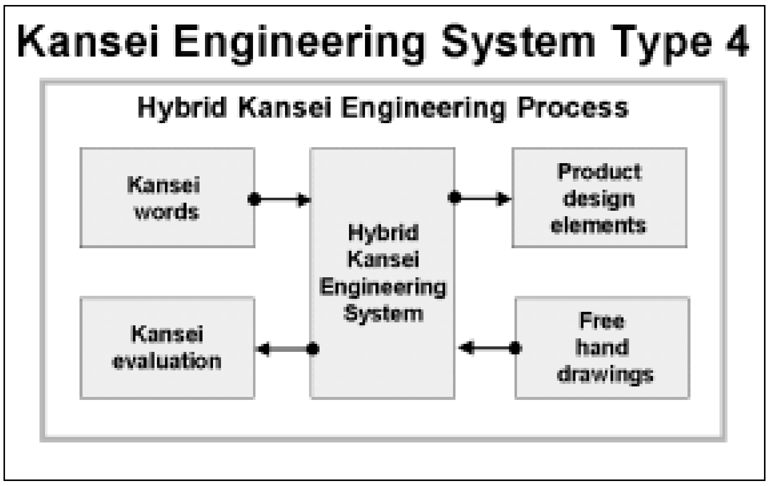

Kansei Engineering System Type 4
Hybrid Kansei Engineering Process
Kansei words
Hybrid Kansei Engineering System
Product design elements
Kansei evaluation
Free hand drawings

Figure 25: Components of a hybrid Kansei Engineering System (Nagamachi 1999)

### 3.5.3.5 Type 5: Virtual Kansei Engineering

This technique is new and combines KE and Virtual Reality (VR). The advantage of VR is that it enables people to experience computer generated virtual designs by means of a head-mounted display. Control is by means of data gloves. With this technique customers can evaluate the new product that were built by means of KE. Nagamachi constructed a Virtual Kansei Engineering kitchen design system for Matsushita. The kitchen was designed by KE to fit the exact customer needs. Subsequently the customer could evaluate the virtual kitchen by means in terms of his specific Kansei requirements.

## 3.5.4 Main Kansei Engineering steps

The following main steps are used in a typical KE process. It is a category classification method and is used most often today.

- Clearly define the product purpose.
- Collect the Kansei data using various marketing methodologies.

- Determine the Kansei product mission or baseline. This is the main purpose of the product. In the case of AEDES this could have been "Total Architectural Knowledge Management".
- Break the base line product concept down into primary, secondary and tertiary sub-concepts.
- During the breakdown pay special attention to the appropriate design metrics and issues such as size, mass and material.
- Implement tests such as ergonomic engineering in order to find more detailed design specifications.
- Summarise the overall specifications and review whether they fit the baseline concept.
- Verify the results with the designers' 3D design mock-ups.
- Make adjustments to the final requirements.

### 3.5.5 The Semantic Differential Method

Advertising and marketing men are frequently faced with the problem of quantifying subjective data with regards the reactions of customers to image of a brand, product or company. In an attempt to solve this problem Snider and Osgood (Snider *et al.* 1957) devised a technique called the Semantic Differential technique (SD). SD attempts to measure what meaning a concept have for people in terms of dimensions which have been empirically defined and factor-analysed. There is a remarkable similarity between this technique and Fuzzy sets (Zadeh *et al.* 1970).

Osgood used a seven-point, equal-interval ordinal scale. These scales were usually selected from 50 pairs of polar adjectives. An example is:

| | | | | | | |
|---|---|---|---|---|---|---|
| Good | | | | | | bad |

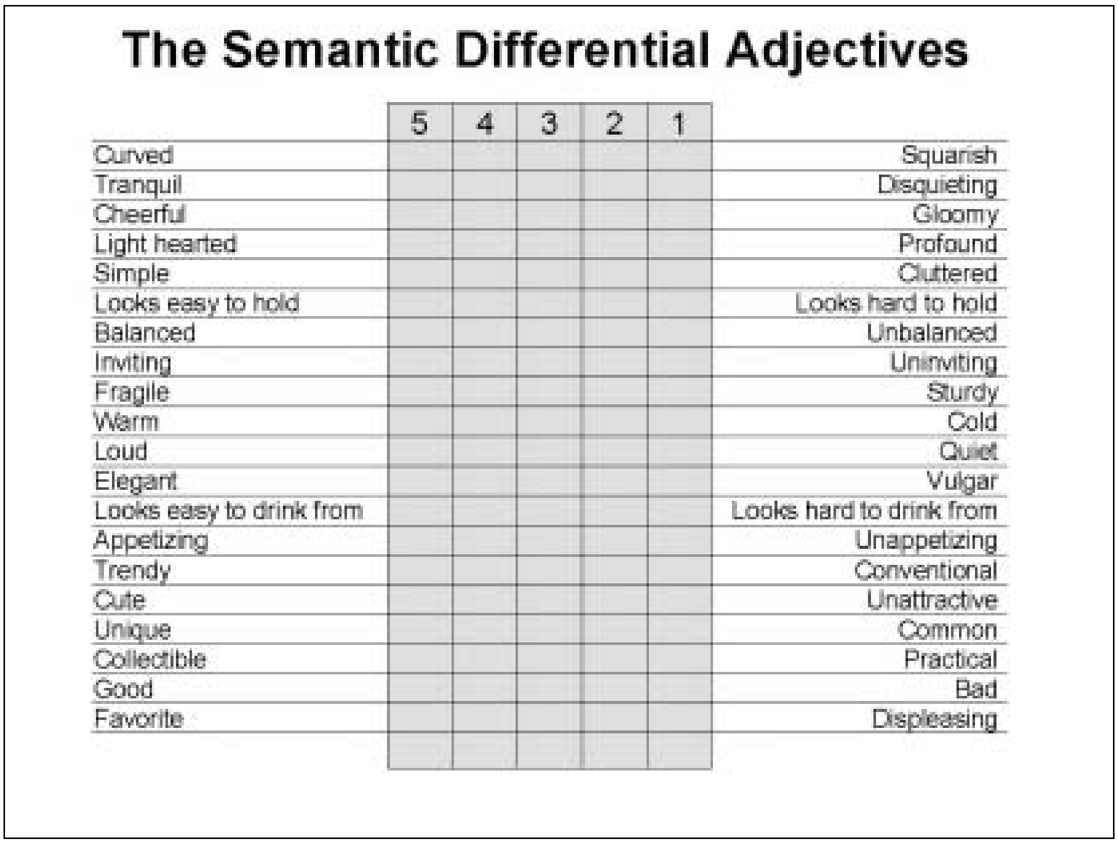
Progressing from left to right on the scale, the positions are described as representing *extremely good*, *very good*, *slightly good*, *being both good and bad*, *slightly bad*, *very bad* and *extremely bad*.

Numeric weights can be assigned to each position. These can be converted to individual or group means and presented in a profile form. The reliability of this method is reasonably high.

The main advantages of SD are:

- It is a quick and efficient way of quantifying large data samples. It captures the direction and intensity of opinions and attitudes towards a concept.
- It provides a comprehensive picture of the image or meaning of a product or personality.
- It is a standardised technique for capturing the multitude of factors which a brand or product comprises.
- It is easily repeatable and reasonably reliable. It can be used on a continuous basis to capture changes in customer attitudes.
- It avoids stereotyped responses and allows individual frames of reference.
- It eliminates some of the problems of question phrasing such as ambiguity and overlapping of statements.

Nagamachi simplified the original Osgood seven-point scale somewhat when he applied it to the design of coffee cups.



**The Semantic Differential Adjectives**

| | 5 | 4 | 3 | 2 | 1 | |
|---|---|---|---|---|---|---|
| Curved | | | | | | Squarish |
| Tranquil | | | | | | Disquieting |
| Cheerful | | | | | | Gloomy |
| Light hearted | | | | | | Profound |
| Simple | | | | | | Cluttered |
| Looks easy to hold | | | | | | Looks hard to hold |
| Balanced | | | | | | Unbalanced |
| Inviting | | | | | | Uninviting |
| Fragile | | | | | | Sturdy |
| Warm | | | | | | Cold |
| Loud | | | | | | Quiet |
| Elegant | | | | | | Vulgar |
| Looks easy to drink from | | | | | | Looks hard to drink from |
| Appetizing | | | | | | Unappetizing |
| Trendy | | | | | | Conventional |
| Cute | | | | | | Unattractive |
| Unique | | | | | | Common |
| Collectible | | | | | | Practical |
| Good | | | | | | Bad |
| Favorite | | | | | | Displeasing |

Figure 26: Adjectives applicable to coffee cups when using the semantic differential method (Nagamachi 1999)

### 3.5.5 Conclusion

KE is used in diverse industries such as the automobile, apparel, home appliance, office machinery, home and cosmetics. These industries include some of the most important car manufacturers in the world. Other applications include diverse fields such as digital colour expression, language analysis, video camera and discomfort analysis by means of cross-modality matching. During a workshop attended by the author, Nagamachi stated that although KE is a very comprehensive system it is not a design system, but rather a design support system. KE could certainly be useful to move certain tacit data down to explicit levels where it could be used in design knowledge management and packaging. It is interesting to note the similarities between SD and the dynamic linguistic variables discussed under 3.2.5.7

## 3.6 Quality Function Deployment (QFD)

### 3.6.1 Introduction

Due to the important role that a holistic approach to quality plays in the product realisation process this section was included. If architectural design knowledge is to be successfully packaged for use during the product life cycle then quality must form an integral part of it. Figure 2 gives an indication of the tacit level of knowledge that QFD operates at. In the precedent system AEDES (Conradie *et al.* 1999) QFD was made an integral part of the briefing and design system. In the present study QFD was used to analyse the needs of the construction team for a very large construction project. This indicated the need for a design processor clearly. It is now known that QFD is too elaborate for the normal architectural design project. For this reason ARGOS does not form a core part of ARGOS anymore. If it is necessary to use it, it should rather be applied as an outside process. Certain projects might still be suitable for QFD.

The author first learned about Quality Function Deployment (QFD) in May 1998. During further study it was discovered that QFD is one of the most powerful and robust methods to support the product design process. Although QFD was already conceived in Japan in the late 1960s the western world was slow to adopt it. At the moment it is still largely unknown in South Africa and met with scepticism. QFD is an adaptation of some of the Total Quality Management (TQM) tools. The author gained practical experience in the use of QFD during a recent Voice of Customer (VOC) exercise with the members of the professional team to establish accurate user requirements for the AEDES prototype system that is a precedent for the present study.

After World War II statistical quality control (SQC) was introduced to Japan and became the central quality activity in the area of manufacturing. This was integrated with the teachings Juran and Ishikawa.  This gradual evolution was strengthened by 1961 publication of Total Quality Control by Feigenbaum. The result of this was that SQC was transformed into TQC during the transitional period between 1960 and 1965. It was at this stage that Akao (1997: 19) became prominent and influential in the subsequent development. Two important factors led to QFD, as we know it today:

- People started to recognise the importance of design quality.
- Companies were already using Quality Control (QC) charts. However the charts were produced during manufacturing after the new products were conceived clearly leaving a quality gap at the initial product conceptualisation.

Akao (1997:19) states that by the time design quality is determined, there should already exist critical Quality Assurance (QA). In 1972 abovementioned deficiencies were addressed in an approach described as “*hinshitsu tenkai*” (Quality Deployment). This established a method to deploy, prior to production start-up, the important quality assurance points needed to ensure the design quality throughout the production process. The method was still inadequate in terms of setting the design quality. This was resolved by means of the quality chart used at the Kobe shipyards of Mitshubishi Heavy Industry. Value Engineering (VE) also influenced QFD. VE is a way to define functions of a product.

In 1978 the term Quality Function Deployment became firmly entrenched. QFD is a literal translation of the Japanese words “*hinshitsu kino tenkai*”.

The most important contributions of QFD are (Akao 1997):

- Established quality management in product development and design.

- Provides a communication tool to designers. Engineers and hopefully architects in future are positioned halfway between the market and production. They need to lead product development. QFD gives a powerful means to build a system for product development.
- QFD can significantly contribute to the software industry.
- Future TQM will become important in future to align company-wide activities to customer focus. Akao believes that Voice of Customer (VOC) should be the common bedrock for creating a partnership of such activities.

The Americans are attempting to combine many different ideas with QFD. This includes TRIZ, Taguchi methods and conflict management.  QFD and Taguchi methods are gaining attention in the USA as effective methods for concurrent engineering. In product and manufacturing process design, a key optimisation tools is Taguchi's Robust Design Method. The prioritisation capabilities of QFD assist the development team to decide where to apply Taguchi's methods.

The global use ISO 9000 series influenced quality control greatly. It established a global quality standard for the first time. The ISO 9000 series require companies to earn their customer's trust by demonstrating a system of quality assurance. ISO defines a quality system as the organisational structure, responsibility, procedure, process and resource for implementing quality control. Akao predicts that QFD will be recognised as an international standard and be incorporated in ISO.

### 3.6.2 What is QFD?

QFD is a method for structured product planning and development that enables a development team to specify clearly the customer's wants and needs and then to evaluate each proposed product or service capability systematically in terms of its impact on meeting those needs (Cohen 1995).

The QFD process involves constructing one or more matrices, sometimes referred to as quality tables. The first of these matrices is called the House of Quality (HOQ) (Figure 27). It displays the customer's requirements along the left and the development team's technical response to meeting those needs along the top. The matrix consists of several sections or submatrices joined together in various ways that contains interrelated information.

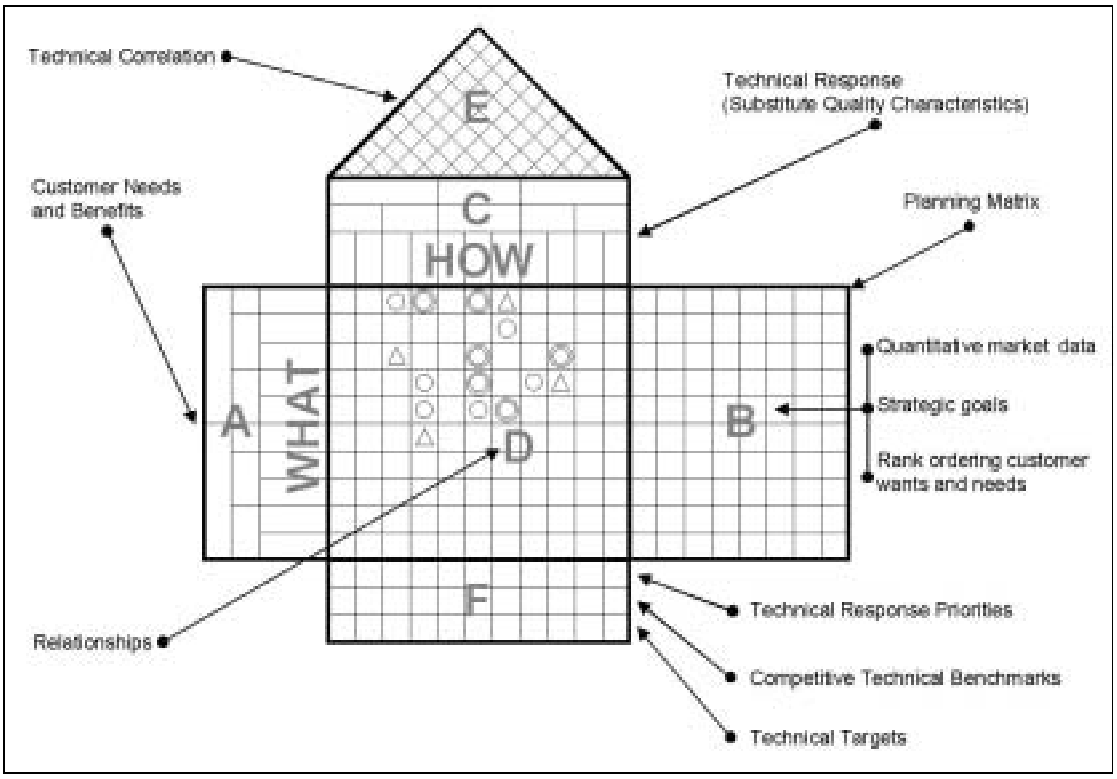

Technical Correlation
E
Technical Response (Substitute Quality Characteristics)
Customer Needs and Benefits
C
HOW
Planning Matrix
Quantitative market data
Strategic goals
Rank ordering customer wants and needs
A
WHAT
D
B
F
Technical Response Priorities
Relationships
Competitive Technical Benchmarks
Technical Targets

Figure 27: Schematic representation of the QFD House of Quality (Cohen 1995:12)

Each of the labelled sections is a structured expression of the product or process development team's understanding of an aspect of the overall planning process for a new product, service or process.

Section *A* contains a structured list of customer wants and needs. The structure is usually determined by qualitative market research. The data are in the form of a tree diagram that is obtained by methods such as a Voice of Customer exercise.

Section *B* contains three main types of information:

- Quantitative market data. This category consists of three columns that indicate *importance to the customer*, *customer satisfaction performance* and *competitive satisfaction performance*.
- Strategic goal setting for the new product or service. This category indicates the level of customer performance being aimed for and the improvement ratio required. The two columns normally used here are *goal* and *improvement ratio*.
- A computation for rank ordering the customer wants and needs. Under this main category the ability to sell the product or service, overall importance to the development team of each customer need and cumulative normalised raw weights are normally captured. These three columns are called *sales point*, *raw weight* and *normalised raw weight.*

Section *C* contains in technical language the description of the product or service they intend to develop. This is normally generated or deployed from the customer wants and needs in section *A*. It is important to note that there will probably not a one-to-one correlation between the user requirements and the technical solutions offered.

Section *D* contains the development team's judgements of the strength of the relationship between the items in *A* and the technical response in *C*. Typically a 9,3,1 scale is used that can also be written by means of special symbols (Figure 27).

Section *E* contains the technical development team's assessment of technical correlation (roof of the quality house) between the items in the technical response.

Section F contains three types of information:

- The computed rank ordering of the technical responses, based on the rank ordering of customer wants and needs from section B and the relationships in section D.
- Comparative information on the competition's technical performance.
- Technical performance targets.

Not all the various sections of the QFD diagram will always be used. In the literature many different variations exist. It is possible to go beyond the initial House of Quality. In the AEDES prototype a system of 5 matrices were used. In this system the HOW of one level of matrix becomes the WHAT at the next level. The effect is that progressive refinement is attained until the desired level of detail is reached.

### 3.6.3 The affinity diagram

The affinity diagram is a means of for organising qualitative information. The hierarchy is built from the bottom up. The source of ideas into the affinity diagram can be internal or external. The team developing the diagram brainstorms internal ideas. Brainstormed ideas are appropriate for a team that has no data to begin with. In the case of the AEDES Voice of Customer (VOC) exercise an extensive prior literature study was made as to what the requirements might be and also to prepare a structured questionnaire to assist the interviewers in asking the correct questions.

Methods that could be used to hear the VOC are (Technosolve 1998; Cohen 1995):
- Focus group interviews.
- Contextual inquiry.
- Conference room interviews.
- Surveys.
- *Gemba* visits (Observe user in his working environment).
- Walk mile in his shoes.
- Customer complaints.
- Customer requests for existing product enhancement or new products.
- Expert opinion.
- Published sources.
- Social events such as parties or exhibitions.

Table 5: Methods of obtaining Voice of Customer (Collated by author)

| | **Formal methods** | | | | **Informal methods** | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **Focus group** | **Contextual inquiry** | **Conference room interview** | **Surveys** | **Gemba visits** | **Walk mile in his shoes** | **Customer complaints** | **Customer requests** | **Events such as exhibitions** |
| **Speed (S,M,F or ~)** | **F** | **S** | **M** | **~** | **S** | **S** | **F** | **S** | **S** |
| **Cost (L,M,H or ~)** | **L** | **H** | **M** | **~** | **H** | **H** | **L** | **L** | **M** |
| **Requirement yield (L,M,H or ~)** | **L** | **H** | **M** | **~** | **M** | **M** | **L** | **~** | **L** |

External ideas are the facts that the team acquires. One of the most thorough methods of obtaining data is by means of a customer interview VOC exercise. This takes very careful planning. Questions are prepared beforehand to prepare the interviewers for their task. During an interview attempts are made to uncover as many real unexpressed customer needs as possible. The complete interview is tape-recorded for a subsequent verbatim transcription.

The AEDES development team conducted eight one-hour interviews with the professional team of a large construction project. The example passage below is an example of such a fragment of information that was communicated by one of the interviewees.

In the subsequent analysis the transcribed interviews are very carefully screened to find passages, statements or remarks that clearly express useful thoughts of the customer. The analysis team then hypothesised as to what the statement actually meant and extracted hypothesised user requirements. In the case of the AEDES VOC exercise 173 such user requirements (URs) were identified. On the analysis form careful note was taken of direct product features that are mentioned during the interview that gives direct clues as to desirable product features.

An example of a probing question during a recent VOC interview was:

**Q. *" So you can store the process case that you had here and transfer it. From that point of view we already have a bit of a case library that you can draw on in the future?"***

A. *"That is how we go from project to project. I've got a file down there, that has got your process methods and production rates and durations, so that every time you come onto a project…e.g. how many bricks do a bricklayer lay, I've got a set standard that is there. I program the whole programme. I know exactly what should be the duration, then the construction guys go along, they program the whole program and if it is way out, I can say to the guys you are smoking yourselves and it is not from my brain, it is from the files, from the database, it is from the cases that we have put together. That case study is somehow static, however, there is suddenly a new way of laying bricks, then the bricklaying process will change, the duration will change and we will have to update that knowledge that information* *will have to change. But, it is probably quite static at the moment. No new techniques have been developed the last couple of years for brick laying."*

During the subsequent analysis of abovementioned verbatim that clearly expresses a customer need the following hypothesised customer requirements and issues and factors were extracted from the transcription.

Issues/ factors:

- Pre-packaged case histories (experiential knowledge).
- Base information across life cycle process.
- Updating of cases.
- Validation of decisions.

Customer Requirement:

- Enables all team members to refer back to and retrieve experiential project knowledge at any level of grain as and when required.

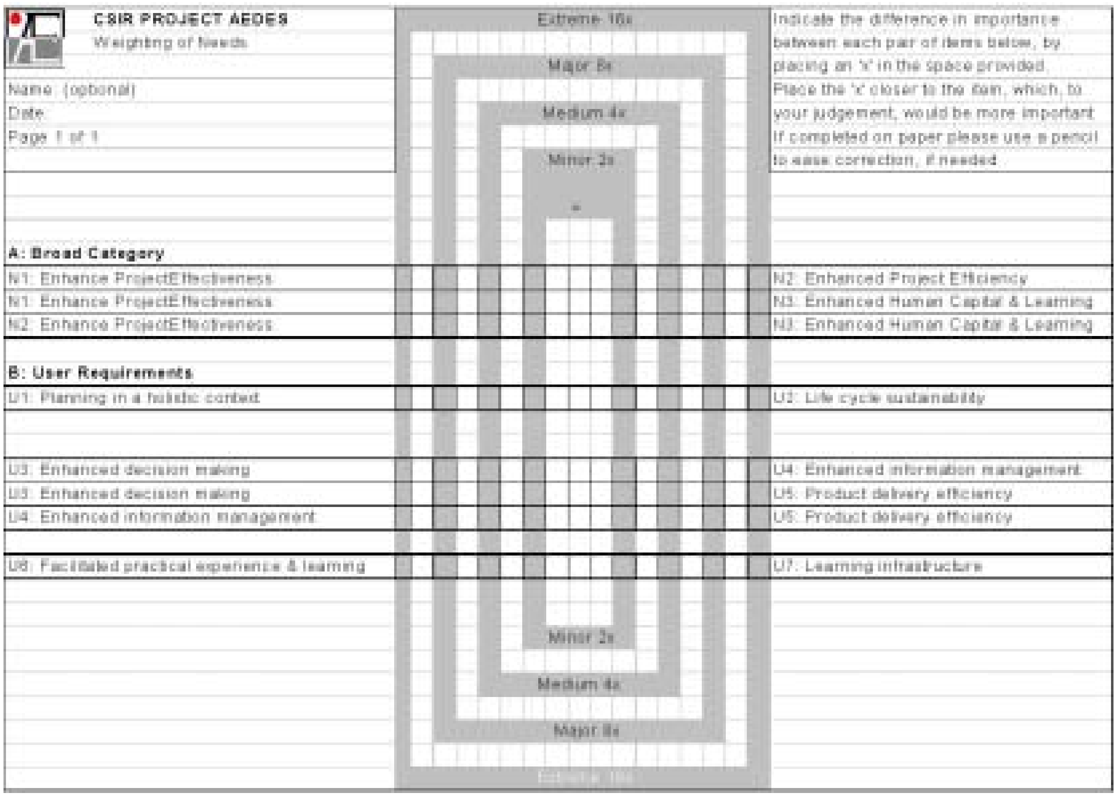
The URs were printed on large sticky labels and pasted onto post-it-notes. The cards were placed on the boardroom table where they could be seen by the entire team. At this stage the team observes total silence and first reads all the cards in sequence in order to get the contents in short term memory. Each member then begins to move the cards together that they think belong together. It is important that the cards are not grouped by similar wording, but rather by similar benefits to the customer. If a card continues to shuttle between different piles the card can be duplicated and placed in two piles. It eliminates a test of wills between two team members.

After the silent sorting process is complete, discussion may start again. The initial 173 URs were grouped into heaps of similar user benefits and a summary title added that best describes the contents of the heap. At this stage the team had 23 abstracted heaps. The next day more effort was put into the grouping exercise and this produced seven higher level URs. At top level it was possible to abstract this to three ultimate user requirements (Table 6).

Table 6: Essential user requirements extracted for the AEDES VOC exercise

| Broad Category | User Requirement | Detailed user requirements |
|---|---|---|
| **Enhance project Effectiveness**<br>(Relates to strategy: doing the right thing) | 1. Planning in holistic context | 1.1 Requirements & methods of life cycle process<br>1.2 Fast track operational processes<br>1.3 Contextual visualisation |
| | 2. Life cycle sustainability | 2.1 Optimise project in terms of sustainability<br>2.2 Timeous planning according to type & scale<br>2.3 Demonstrated financial risk & return |
| **Enhanced project efficiency**<br>(Relates to productivity: doing things right) | 3. Enhanced decision making | 3.1 Sound decision-making<br>3.2 Co-ordinated decision making<br>3.3 Trace-ability & validation of decision |
| | 4. Enhanced information management | 4.1 Access to information<br>4.2 Information flows and interchange-ability<br>4.3 Transparency of interactions & information |
| | 5. Product delivery efficiency | 5.1 Problem solving<br>5.2 Product performance characteristics<br>5.3 Contractor supply chains<br>5.4 Quality assurance<br>5.5 Efficiency in terms of time/cost savings |
| **Enhanced human capital and learning** | 6. Facilitates practical project experience | 7.1 Skills transfer & job creation<br>7.2 Practical training & learning |
| | 7. Learning infrastructure | 7.1 Organisational memory<br>7.2 Experiential project knowledge<br>7.3 Rapid access to experiential knowledge<br>7.3 Generic briefing templates |

Table 7 : Sample of  form used to extract constant sum paired comparisons from users



**CSIR PROJECT AEDES**
Weighting of Needs

Name: (optional)
Date
Page 1 of 1

Indicate the difference in importance between each pair of items below, by placing an 'x' in the space provided. Place the 'x' closer to the item, which, to your judgement, would be more important. If completed on paper please use a pencil to ease correction, if needed.

Extreme 16x
Major 8x
Medium 4x
Minor 2x
=
Minor 2x
Medium 4x
Major 8x
Extreme 16x

**A: Broad Category**

| | |
|---|---|
| N1: Enhance ProjectEffectiveness | N2: Enhanced Project Efficiency |
| N1: Enhance ProjectEffectiveness | N3: Enhanced Human Capital & Learning |
| N2: Enhance ProjectEffectiveness | N3: Enhanced Human Capital & Learning |

**B: User Requirements**

| | |
|---|---|
| U1: Planning in a holistic context | U2: Life cycle sustainability |
| U3: Enhanced decision making | U4: Enhanced information management |
| U3: Enhanced decision making | U5: Product delivery efficiency |
| U4: Enhanced information management | U5: Product delivery efficiency |
| U6: Facilitated practical experience & learning | U7: Learning infrastructure |

Subsequently a method called "constant sum paired comparisons" was used where each attribute is compared to every other attribute (Cohen 1995:97). This technique is preferred over a five point scoring scales because it yields better statistical accuracy and is non-ordinal. The QFD institute in Detroit also recommends this technique. In this technique a participant is expected to weigh up pairs of factors against each other. This takes careful thinking. The comment is normally that unlike factors are compared, however it must be seen as an importance rating or a rating that tries to determine which factor gives you the most problems. Of the 35 questionaires sent out, a total of nine were returned.

The results of the pairings were analysed and the results indicated clearly the need for a strategic what-if scenario system across the project life-cycle.

The consolidated results produced the results below that are ordered from the most important to the least important.

Table 8: Relative importance of user requirements within group

| User requirement | Level of Consensus | Relative importance as a percentage | Normalised relative importance |
|---|---|---|---|
| U2 – Strategic what-if scenarios across project life cycle | Good | 30.91 | 0.31 |
| U1 – Planning in an appropriate holistic context | Good | 24.89 | 0.25 |
| U6 – Practical project experience and learning | Good | 16.03 | 0.16 |
| U7 – Learning support infrastructure during project delivery | Useful | 11.47 | 0.11 |
| U5 – Project delivery efficiency | Useful | 7.35 | 0.07 |
| U4 – Information management across project life cycle | Useful | 7.33 | 0.07 |
| U3 – Decision management across project life cycle | Good | 2.02 | 0.02 |

The *user requirements* (Customer needs and benefits) can now be filled in on the What side of the QFD matrix. The *relative importance* values can be placed on the planning matrix. The technical team can proceed to generate technical concepts (Technical response or substitute quality characteristics). Analysis of the most important requirements clearly indicates that the following functionalities are required:

- Structured data in appropriate classification containers
- Life cycle software tool modularity
- Systems approach
- Data hierarchies because construction element relationships exhibit a hierarchical structure.
- World wide web connectivity of tools and data
- Data labelling
- Desktop working environment
- Life cycle  supply support data like material performance libraries
- Learning support such as intelligent archived case studies that is a complete cognitive snapshot of the various design factors.
- Process support

From this it becomes clear that life cycle process and data integration need to be created. This must further be supported by appropriate modular analysis tools and packaging of designs at various levels of granularity. This is a clear indication that a need for intelligent Case-based Reasoning enabled components exists that could assist the project team with design and operational decisions. This type of component must be flexible to operate in many different software environments at various stages of the product life cycle.

### 3.6.4 Kano's model of user satisfaction

The Japanese TQM consultant Noriaki Kano provides useful insights of customer satisfaction as it relates to product characteristics. Kano's model divides product characteristics into three distinct categories, each of which affects customers in a different way. The three categories are (Figure 28):

- *Dissatisfiers*. These are "must-be", "basic" or "expected" characteristics.
- *Satisfiers*. These are also known as "one-dimensional" or "straight-line" characteristics.
- *Delighers*. These are also known as "attractive" or "exciting" characteristics.

#### 3.6.4.1 Dissatisfiers

These are product characteristics that the customer takes for granted when they are present, but that cause dissatisfaction when they are missing. Dissatisfiers are things customers do not normally ask for, because they tacidly assume that they will be taken care of. If a product or service is delivered that has many dissatisfiers, customers will be extremely unhappy.

If dissatisfiers are eliminated then customers will hardly notice all the work that has been done to eliminate the dissatisfiers. The reduction of dissatisfiers can only raise customer satisfaction to a "not dissatisfied" state.

#### 3.6.4.2 Satisfiers

This is a feature or characteristic that a customer wants in his product and would usually ask for. The more satisfiers that are provided, the happier customers will be. It is also known as desired quality because it represents the aspects of the product that define it for the customer. Examples of this that were expressed during the AEDES VOC exercise are:

- Compatible data interchange amongst project participants.
- Project specific configuration of software.
- On-line availability of information

In the competitive world of software development one can expect satisfiers to be present in all the competitive products.

#### 3.6.4.3 Delighters

These are product attributes or features that are pleasant surprises to customers when they first encounter them. However if delighters are not present, customers will not be dissatisfied, because they do not know what they are missing. We cannot learn about product delighters by directly asking our customers. Examples of delighters are not as instructive as examples of satisfiers and dissatisfiers. Each delighter is unique and no particular patterns can be identified. Some delighters are entire new products that created entirely new markets. In the present study delighters for the portable design cases called ARGOS could be:

- Integration into any ActiveX compliant container environment such as spreadsheets.
- All pertinent design information available in a convenient to use environment of the users choice.
- Support for design via the Internet.
- Integration of function, shape and quality into a single highly portable mini-case environment.

- The ability to use structured information from the past to assist with future design problems.

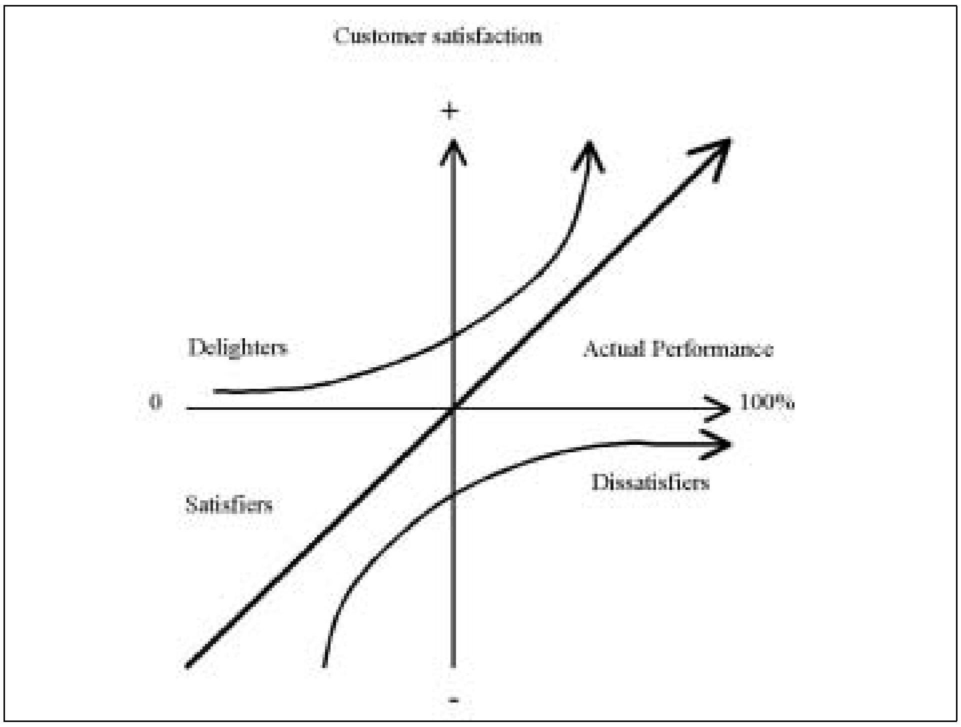


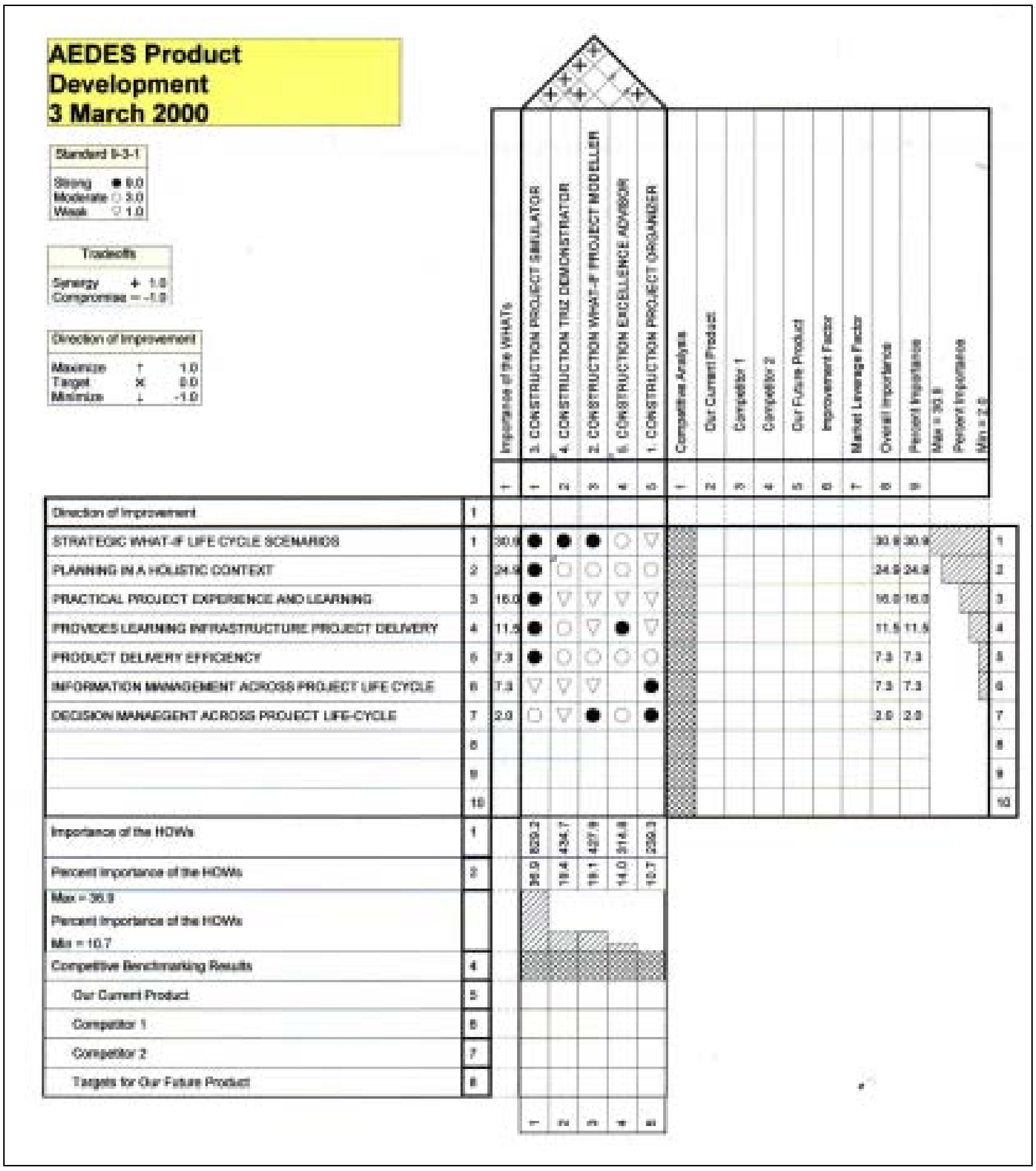
Figure 28: Kano's customer satisfaction diagram (Cohen 1995:37)

The needs that delighters fill are often called latent or hidden needs because they are not directly communicated. QFD offers some assistance in this regard where the interviewers during a VOC exercise attempts to scaffold into the unconscious product desires of the client. During subsequent analysis of the VOC the analysts attempt to cover assumed, expressed and latent elements. QFD is particularly useful because it helps the development team to clearly separate customer needs from technical solutions.

### 3.6.5 QFD software

Very good commercial QFD software such as QFD/Capture Professional is available. Typical features of this software includes:

- The ability to publish HTML web page output of QFD reports.
- Generate customer surveys in text, Microsoft Word, Rich Text Format and HTML Web Page formats.
- Produce market opportunity map reports identifying the best opportunities for product improvement.
- Generate relationship tree diagrams showing measures for each requirement in a graphical tree and branch format.
- Print out and work with blank chart templates, which are useful as documents-in-progress during team meetings.



AEDES Product Development
3 March 2000

| Standard 9-3-1 | | |
|---|---|---|
| Strong | ● | 9.0 |
| Moderate | ○ | 3.0 |
| Weak | ▽ | 1.0 |

| Tradeoffs | | |
|---|---|---|
| Synergy | + | 1.0 |
| Compromise | − | -1.0 |

| Direction of Improvement | | |
|---|---|---|
| Maximize | ↑ | 1.0 |
| Target | ✕ | 0.0 |
| Minimize | ↓ | -1.0 |

| | | Importance of the WHATs | 3. CONSTRUCTION PROJECT SIMULATOR | 4. CONSTRUCTION TRIZ DEMONSTRATOR | 2. CONSTRUCTION WHAT-IF PROJECT MODELLER | 5. CONSTRUCTION EXCELLENCE ADVISOR | 1. CONSTRUCTION PROJECT ORGANIZER | Competitive Analysis | Our Current Product | Competitor 1 | Competitor 2 | Our Future Product | Improvement Factor | Market Leverage Factor | Overall Importance | Percent Importance | Max = 30.9 Percent Importance Min = 2.0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | | |
| Direction of Improvement | 1 | | | | | | | | | | | | | | | | | |
| STRATEGIC WHAT-IF LIFE CYCLE SCENARIOS | 1 | 30.9 | ● | ● | ● | ○ | ▽ | ▨ | | | | | | | 30.9 | 30.9 | ▨ | 1 |
| PLANNING IN A HOLISTIC CONTEXT | 2 | 24.9 | ● | ○ | ○ | ○ | ○ | ▨ | | | | | | | 24.9 | 24.9 | ▨ | 2 |
| PRACTICAL PROJECT EXPERIENCE AND LEARNING | 3 | 16.0 | ● | ▽ | ▽ | ▽ | ▽ | ▨ | | | | | | | 16.0 | 16.0 | ▨ | 3 |
| PROVIDES LEARNING INFRASTRUCTURE PROJECT DELIVERY | 4 | 11.5 | ● | ○ | ▽ | ● | ▽ | ▨ | | | | | | | 11.5 | 11.5 | ▨ | 4 |
| PRODUCT DELIVERY EFFICIENCY | 5 | 7.3 | ● | ○ | ○ | ○ | ○ | ▨ | | | | | | | 7.3 | 7.3 | ▨ | 5 |
| INFORMATION MANAGEMENT ACROSS PROJECT LIFE CYCLE | 6 | 7.3 | ▽ | ▽ | ▽ | | ● | ▨ | | | | | | | 7.3 | 7.3 | ▨ | 6 |
| DECISION MANAEGENT ACROSS PROJECT LIFE-CYCLE | 7 | 2.0 | ○ | ▽ | ● | ○ | ● | ▨ | | | | | | | 2.0 | 2.0 | | 7 |
| | 8 | | | | | | | ▨ | | | | | | | | | | 8 |
| | 9 | | | | | | | ▨ | | | | | | | | | | 9 |
| | 10 | | | | | | | ▨ | | | | | | | | | | 10 |
| Importance of the HOWs | 1 | | 829.2 | 434.7 | 427.9 | 314.8 | 239.3 | | | | | | | | | | | |
| Percent Importance of the HOWs | 2 | | 36.9 | 19.4 | 19.1 | 14.0 | 10.7 | | | | | | | | | | | |
| Max = 36.9 Percent Importance of the HOWs Min = 10.7 | | | ▨ | ▨ | ▨ | ▨ | | | | | | | | | | | | |
| Competitive Benchmarking Results | 4 | | ▨ | ▨ | ▨ | ▨ | ▨ | | | | | | | | | | | |
| Our Current Product | 5 | | | | | | | | | | | | | | | | | |
| Competitor 1 | 6 | | | | | | | | | | | | | | | | | |
| Competitor 2 | 7 | | | | | | | | | | | | | | | | | |
| Targets for Our Future Product | 8 | | | | | | | | | | | | | | | | | |
| | | | 1 | 2 | 3 | 4 | 5 | | | | | | | | | | | |

Figure 29: QFD/Capture product planning matrix screen (Author)

During the prototype development of the AEDES software the author developed QFD software that could integrate the architectural briefing and design process directly in an underlying database. This was an attempt to make the information captured during the QFD briefing and design sessions directly available to other distant members of the design team. The biggest difference between the AEDES QFD and standard software is the fact that it worked in depth as well. In-depth implied that the user could see more detail by clicking on the intersection of a specific row or cell (Figure 31).

QFD has a practical limitation in the sense that it cannot conveniently accommodate more than a 20 by 20 matrix. Architecture contains information at many different levels of detail that is likely to give rise to very large matrices. This was solved by means of the in-depth method. The disadvantage of the latter is that it is not possible to see information directly at a glance. However reports were developed that can be printed out and studied at leisure.

The in-depth method required a special database structure that is detailed in Figure 31. The relational database table *QFD* contained the main QFD project information. Two main hierarchies branch from this main table, i.e. the *QFDHow* and *QFDWhat* branches each having respective subtables called *QFDSubHow* and *QFDSubWhat*. Special connecting tables (somewhat unusual in relational database design) were used to keep book of the relationships between the What and How data branches. The relationships are what would occur in section D (Figure 27). The tables *QFDRoof* and *QFDSubRoof* are self-referring tables (recursive) and are designed to support the relationships that are required by the QFD technical correlations (QFD roof).

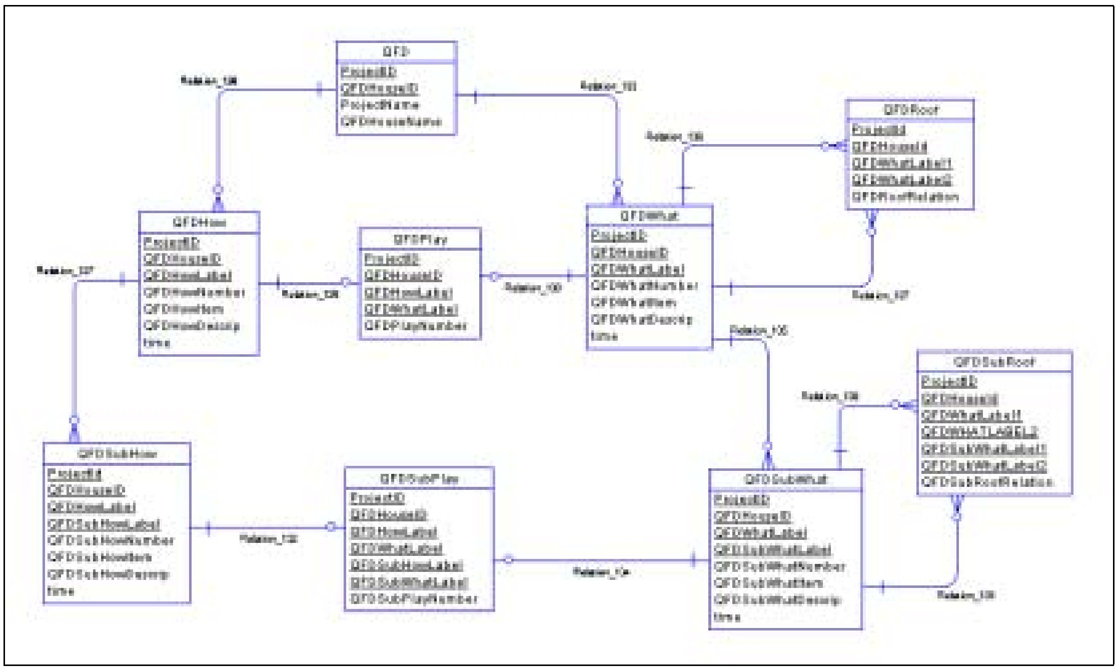
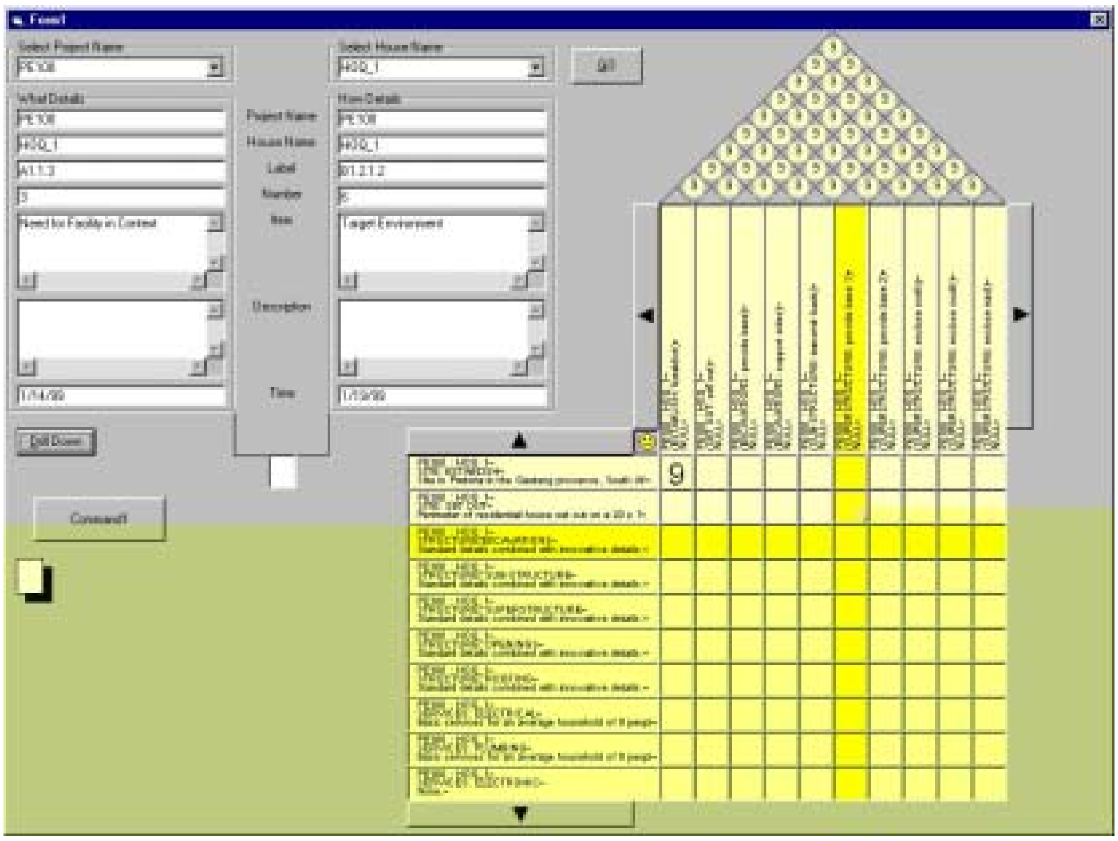


Figure 30: Relational database tables used in the AEDES prototype QFD software (Author)

The software only allowed viewing of the design data. Editing was accomplished by means of special database forms. The QFD software provided a convenient means to view the numerous different technical correlations that exists in architecture. The software provided convenient navigational command buttons that facilitated navigation across a larger than displayed virtual QFD matrix. A drawback of this was that it was not possible to view the entire matrix at a glance. The author is of the opinion that it is not always necessary to see all design issues at once in architecture, because not all design factors at all levels are so closely related that it is necessary to have simultaneous visual display. Future improvement of the software could be to write QFD software in a Visual Basic ActiveX control. This would greatly improve the usefulness of the QFD software because it would then be possible to use the advanced methodology in a convenient environment such as a spreadsheet, CAD systems or it could be integrated into software shells developed in languages such as Visual Basic or Visual C++.



| | What Details | | How Details |
|---|---|---|---|
| Select Project Name | PE108 | Select House Name | HOQ_1 |
| Project Name | PE108 | | PE108 |
| House Name | HOQ_1 | | HOQ_1 |
| Label | A1.1.3 | | B1.2.1.2 |
| Number | 3 | | 6 |
| Item | Need for Facility in Context | | Target Environment |
| Description | | | |
| Time | 1/14/99 | | 1/13/99 |

Figure 31: Typical screen of the AEDES prototype QFD software (Author)

## 3.7 Theory of inventive problem solving (TRIZ)

### 3.7.1 Introduction

There are two groups of problems people face, those with generally known solutions and those with unknown solutions. Those with known solutions can usually be solved by information found in the technical literature or through extensive training. These solutions follow the general pattern of problem solving. Here a standard solution is elevated to a standard problem of a similar or analogous nature. A standard solution is known and from that standard solution comes a particular solution to the problem.

The other type of problem has an unknown solution. It is called an inventive problem and may contain contradictory requirements. In modern times inventive problem solving falls in the field of psychology where the links between the brain, insight and innovation are studied. Methods such as brainstorming and trial-and-error are commonly suggested. Depending on the complexity of the problem, the number of trials will vary. If the solution is within the field of experience then the number of trials will be fewer. If the solution is not found the inventor must look beyond his experience and knowledge to new fields such as manufacturing or aviation. Then the number of trials will grow large depending on how well the inventor can master psychological tools like brainstorming, intuition and creativity. A further problem is that psychological tools like experience and intuition are difficult to transfer to other people in the organisation.

This leads to what is called psychological inertia where the solutions being considered are within the inventor’s own experience and do not consider alternative technologies to develop new concepts. When we overlay the limiting effects of psychological inertia on a solution map covering broad scientific and technological disciplines the ideal solution might lie outside the inventor’s field of expertise. Psychological inertia defeats randomness and leads to looking only where there is personal experience.

### 3.7.2 TRIZ

Genrich S. Altshuller, born in the former Soviet Union in 1926, developed a superior approach relying on technology. His curiosity about problem solving led him to search for standard methods. Altshuller screened over 200 000 patents looking for inventive problems and how they were solved. Only 40 000 had somewhat inventive solutions, the rest were straightforward improvements. At this stage it is estimated that more than a 1 000 000 patents have been screened world-wide. Altshuller defined an inventive problem as one in which the solution causes another problem to appear. Usually inventors must resort to a trade-off and compromise between the features and thus do not achieve an ideal solution. In his study of patents he found that many described a solution that eliminated or resolved the contradiction and required no trade-off. Altshuller identified five levels of inventive solutions (Kaplan 1996:2; Mazur 2001):

- Level one. These are routine design problems solved by methods well known within the speciality. No invention is required. About 32% of the solutions fell into this level.
- Level two. These are solutions that leave the existing system fundamentally unchanged. New features are introduced or minor improvements are made to the existing system. This is effected by known methods and sometimes compromises may be made. About 45% of the solutions fell into this level.
- Level three. This constitutes an essential improvement of an exiting system. Methods outside the known industry are used. Certain contradictions need to be resolved. About 18% of the solutions fell into this category.

- Level four. At this level inventions are characterised by solutions found in more in science than in technology. Only about 4% of the solutions fell into this category.
- Level five. This is the level where rare scientific discoveries or pioneering inventions occur. Only about 1% of the solutions fell into this category.

He also noted that with each succeeding level, the source of the solution required broader knowledge and more solutions to be considered before an ideal one could be found. Altshuller found that 90% of the problems engineers faced had been solved somewhere before. If engineers could follow a predictable through the various levels and using their knowledge and experience most of the solutions could be derived from knowledge already present in the particular company or industry.

Altshuller distilled the problems, contradictions and solutions to these patents into a comprehensive theory of inventive problem solving which he named TRIZ.

There are a number of laws in the theory of TRIZ. One of them is the law of Increasing Ideality. A technical system evolves in such direction as to increase its degree of Ideality (Kaplan 1996). Ideality is defined as the quotient of the sum of the system's useful effects, $U_i$, divided by the sum of its harmful effects, $H_j$.

$$Ideality = \frac{\sum U_i}{\sum H_j}$$

Useful effects include all the valuable results of the system's functioning. Harmful effects include undesired inputs such as cost, the space occupied, energy consumed, pollution and danger. The ideal state is one where there are only benefits and no harmful effects also termed the *Ideal Final Result*. From a design point of view, engineers must continue to pursue greater benefits and reduce cost of labour, materials, energy and harmful side effects. If the improvement of a benefit results in increased harmful effects, a trade-off is made, but the Law of Ideality drives designs to eliminate or solve any trade-offs or design contradictions. The ideal final result will eventually be a product where the beneficial function exists but the machine itself does not. The evolution of the mechanical spring-driven watch into the electronic quartz crystal watch is an example of this move towards Ideality.

Boris Zlotin and Alla Zusman, TRIZ scientists at the American company Ideation and students of Altshuller have developed an "Innovative Situation Questionnaire" to identify the engineering system being studied, its operating environment, resource requirements, primary useful functions, harmful effects and ideal result.

## 3.7.3 Steps in using TRIZ

### 3.7.3.1 Formulate the problem: the prism of TRIZ

This first step is to restate the problem in terms of physical contradictions. Identify problems that could occur. Could improving one technical characteristic in solving the problem cause other technical characteristics to worsen, resulting in secondary problems? Are there technical conflicts that might force a trade-off?

### 3.7.3.2 Search for previously well-solved problems

Altshuller extracted from over 1 500 000 worldwide patents 39 standard technical characteristics that cause conflict. These are called the 39 Engineering Parameters. Find the contradicting engineering principles. First find the principle that needs to be changed. Then find the principle that is an undesirable secondary effect. State the standard technical conflict.

### 3.7.3.3 Look for analogous solutions and adapt to solution

Altshuller also extracted from the worldwide patents 40 inventive principles. These are hints that will help an engineer find a highly inventive (patentable) solution to the problem. To find which inventive principles to use, Altshuller created the table of Contradictions. This table lists the 39 Engineering Parameters on the X-axis (undesired result or conflict) and Y-axis (feature to change or improve). The appropriate Inventive principles that could lead to a solution are listed in the intersecting cells.

### 3.7.3.4 Socially responsible TRIZ

Structurally and philosophically TRIZ methods look at the big picture. During problem definition the TRIZ practitioner looks at nine combinations of the past, present and future models of the sub-system, system and super-system. Interactions, resources, harmful and secondary effects are identified during the definition of the problem.

Terninko (1999:285) states that the TRIZ method can support sound environmental design through the recognition of resources within the sub-systems, systems and super-systems. If the future of the system and super-system is well understood then possible future disastrous effects can be avoided. He suggests a different type of TRIZ formula to take account of the harmful effects.

$$Ideality = \frac{\sum benefits}{\sum \cos ts + \sum harms}$$

The equation above is more a construct than a directly usable equation. This particular version of the ideality equation contains cost and harmful effects in the denominator.

It is not difficult to identify the benefits and this is what the TRIZ specialist normally tries to understand. The identification of possible harmful effects and its alternatives is just as important. The product designer is normally far too casual about the costs and harms in the denominator. See item 4.2.1.2 for Sustainable Development.

TRIZ does represent a method that can be socially responsible, but the practitioner must resist pressure from society and industry for rapid and incomplete analysis. Organisations are often driven by profit while ignoring the customer and the medium to long consequences of their solutions.

## Summary

The techniques and product innovation methodologies discussed in this chapter are useful at various tacit and explicit levels. They are also applicable in different building life cycle phases (Figure 2).

It is clear that after the initial optimism about the possibilities of AI in design, a more mature and realistic approach is now followed. CBR is a promising sub-field of AI that can greatly contribute to the contextual storing of design knowledge. It is clear that AI should be used more in the background and especially in architecture automatic adaptation of designs should not be attempted.

Knowledge Management is becoming very prominent although there are still unsolved problems. However many researchers are working on the particular sub-problems due to the importance of this for the global economy. KM is still fluid, however the theory is well understood such as the movements of the knowledge cycle. The sharing of knowledge is important in any enterprise and this is supported by the current importance attached to intellectual capital. *Concept extraction* and *Natural Language Processing* remains problematic, however significant progress has already been made. The current and emerging technical standards that form a barrier to responsive NGM were identified. The main requirements for a KM enabling environment are:

- Communication
- Design team flexibility and responsiveness
- User Interface and information search
- Project resource integration and access

The problems of ontology and the role that AI can play in Knowledge Based Design were investigated. It was observed that CBR and the concept selection cycle of Pugh (1996) bear striking similarities. The various main known problem-solving architectures were investigated and the conclusion can be made that CBR, RBR and MBR should be not be seen in isolation but should rather be viewed as a continuum of techniques.

The use of fuzzy sets as a means of formulating dynamic linguistic variables for aiding the retrieval of design knowledge in general and cases specifically were investigated. It was discovered that the semantic differential method of Snider and Osgood (Snider *et al*. 1957) and the semantic differential adjectives as used by Nagamachi bear a relationship to the approach advocated by the author.

The analysis of the characteristics of manufacturing such as process, flow and throughput indicate that these are not directly applicable to problems under consideration, but should rather be applied at the process level. Concurrent Engineering is an important technique to avoid the so-called time-trap. This is where the life cycle time of products decreased while the time spent on product development greatly increased. Three possible strategies could be identified as CE guiding principles:

- Parallelisation.
- Standardisation
- Integration

The theories of Goldratt showed that the manufacturing environment is particularly applicable for Theory of Constraints and that the system optimum is not the sum of the local optima.

Taguchi techniques indicate the importance of off-line and on-line quality control. This indicates that quality is related to the loss to society caused by a product during its life cycle. In terms of the current thesis these methods should rather be used select appropriate materials to minimise life cycle costs in the context of sustainable development.

The advantages of the fuzzy Front End (FFE) was identified:

- It lasts a long time
- Cheap place to look for cycle time
- Individual companies have big performance differences

The investigation of objects indicates that it is the preferred way to achieve abstraction, generalisation and interaction in systems supporting the life cycle development process. The unique way those objects were used in the precedent systems PREMIS and AEDES were discussed. This section was concluded with strategies followed by Microsoft to establish if these could offer opportunities for the packaging of architectural design and design parameters.

Kansei Engineering (KE) is a mature and useful technique to quantify cognition and product image in such a way as to influence the product development process. KE operates at a very high tacit level is could make a significant contribution to the storage of tacit architectural design information.

The QFD exercise undertaken indicated that the ability to generate what-if scenarios across the project life cycle as the most important user requirement. QFD as a technique to extract raw architectural user requirements was pioneered in the AEDES system. QFD is useful if the time and cost can be justified. QFD is an important technique in the manufacturing industry and was one of the techniques that saved the Detroit automotive industry from ruin in the face of severe competition from Japan. The contribution that QFD can make in architecture is dependent on the acceptance that this slightly elaborate technique can gain.

The chapter is concluded with TRIZ that is a powerful method to solve inventive problems. However the present commercial TRIZ software emphasise engineering type of problems. A significant amount of work will have to be done to make its use tractable in architecture.

# Chapter 4: Precedents to the present research

## Introduction

This chapter provides an overview of the two precedent systems PREMIS and AEDES that the author developed. This provides useful insights as to the approach that should be taken in the present research. It also highlights lessons learnt in the two critiques.

## 4.1 The PREMIS Facilities Management System

### 4.1.1 Introduction

Conradie (1996) developed the PREMIS (Professional Real Estate Management Information System) over the last 12 years. The core system currently has the following main components (Figure 1).

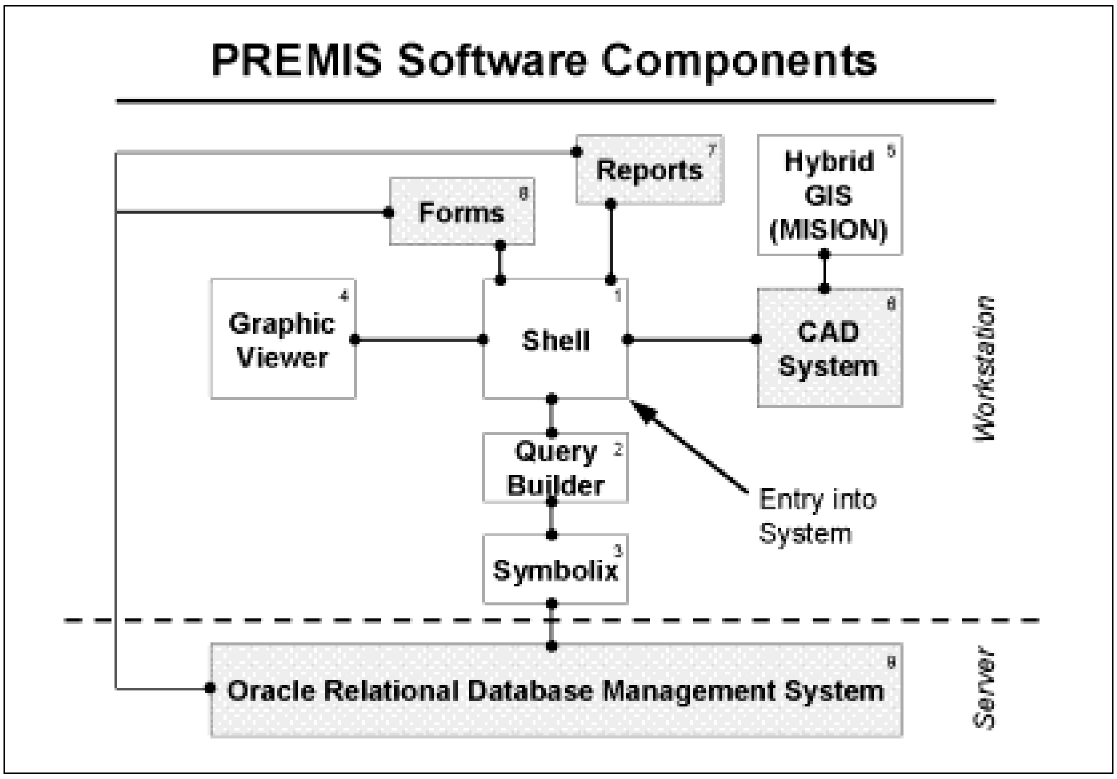


Figure 1: PREMIS Facilities Management Software Components (Author)

The PREMIS software system is based on industry standard components as far as possible. The system uses the Oracle database and AutoCAD as graphics editor. To these industry standard components were added a multi-purpose Ad-hoc query builder, Symbolix graphic visualisation language, viewer and MISION hybrid GIS system. These components are integrated together by means of a fully user configurable shell.

### 4.1.2 Intrinsic design principles

- The system core is generic and can act as a platform for a broad spectrum of facilities management application software.

- Data are organised as information in raw format. No processed information is stored because it is impossible to unscramble scrambled information. The cost per m² will not be stored, rather the number of m² and the cost. The cost/m² is then calculated by means of a report. This enables users and application writers to customise the system easily to suit their specific environment. The system is designed in a generic way and is therefore very flexible and adaptable to changing needs.
- There is a clear distinction between data and reports and ultimately process. This is to ensure a long life system that can easily adapt to changes in the facilities management arena.
- Process is seen as something that sits on top of the data and uses the data. A good example of this is the difference between the way that different organisations might operate their space management systems. The author wrote two distinct systems for two different organisations. Although the reports and processes on top are vastly different they use exactly the same PREMIS/ Oracle database tables and forms. Company A might use the concept of calculated area to derive rent and company B operational cost apportioning.
- Software is designed for a high level of modularity to ensure the replacement of obsolete software components with more modern ones without affecting the stability of existing components (Open ended structure).
- The data backbone of the system is the Property category. This consists of a hierarchy starting from country right down to site, building, floor and space. Unless this structure is used it is impossible to handle the large range of different facility sizes. The property category incorporates the basic data required for asset registers and space management.
- The system is highly optimised for speed. Very high throughput is achieved even on moderate capability equipment due to the PREMIS systems architecture. The *Symbolix* language is a purpose made language that is compiled to achieve high speed. The index system in the MISION, hybrid GIS is highly optimised and shows no deterioration even with large file sizes over 2 Mb. Oracle PRO*C is used to ensure the highest possible communication speed with the database.
- The use of an implicit linking between alphanumeric data in the relational database and the graphic objects was pioneered in this system (Figure 2).

### 4.1.3 A typical application

A typical application such as the Building Maintenance Management System (BMMS) consists of 84 relational database tables and 60 Oracle forms of varying complexity. The main data table categories covered by the system is Property, Legal, People, Elements and Organisation Structure.

The system uses object-oriented principles to link graphic and alphanumeric data. This facilitates the integration of the graphic, alphanumeric and spreadsheet environments.

Implicit linking of graphic data to alphanumeric data is used. By using object-oriented principles the graphic data are linked to the alphanumeric data. This is done in such a way that although the items exist in totally separate environments relationships exist by virtue of the fact that the graphic object names are the same as the concatenated data key fields in Oracle. This has far reaching implications in the sense that one alphanumeric record may have multiple graphic representations. This also ensures a very high level of modularity and portability. Data may be independently added in the two environments. Relationships exist the moment that the graphic object name and the concatenated alphanumeric database record key are the same (Appendix A).

This particular application integrates strategic facilities planning with operational maintenance. This is achieved by means of a generic list of 352 construction elements that can be used for strategic planning. The very same list of elements can be used in an operational environment by means of maintenance action verbs like *Remove*, *Replace*, *Repair*, *Patch*, *Repaint*, *Refix/ Refit*, *Cut*, *Clean* and *Service*. These actions can then be combined to the elements to derive unlimited, but at the same time well-structured, combinations of actions.

The system can expand from a single user system to a large networked system. This ensures an unlimited growth path. The Oracle database performs particularly well with a large amount of data without significant deterioration in speed.

The PREMIS, Oracle forms can be deployed over the web by means of the Oracle web cartridge. Oracle offers a very high level of connectivity and openness to other systems. It is therefore relatively easy to interface Oracle to other systems.

The PREMIS graphic visualisation language SYMBOLIX offers superior graphic symbol processing capabilities. SYMBOLIX enables the programmer to design parametric symbols that vary their shape, colour and size according to parameters or values retrieved from the ORACLE database. SYMBOLIX enables graphic visualisation of issues. This is more advanced than business graphics, because the results can be geographically placed as well.

ORACLE FORMS 4.5 (one of the products in the Oracle Developer/2000 suite of software) is being used as the user interface to enter data into the database. FORMS 4.5 is a generic forms front end that supports ODBC (Open Database Connectivity), Custom Interfaces, User Exits, Visual Basic and OLE (Object Linking and Embedding)

Due to the fact that ORACLE is used as the database a user can start with a single user workstation with Personal Oracle 7 for Windows and expand to a virtually unlimited multi-user size. PREMIS is therefore able to offer a low cost entry point but with no upper limit on the size of the estate. In fact an estate can be anything from a single building to a complete estate consisting of many buildings.

PREMIS is designed in such a way that it facilitates very flexible combinations of multi-user access. It is possible in a multi-user system to install PREMIS on say ten workstations. If the user has only bought a 3-user concurrent license any combination of three out the ten users can used simultaneously. The access control will impose a limit of three concurrent users at any time.

### 4.1.4 Critique of PREMIS

PREMIS was developed over many years and significant experience was gained in the display of large volume data in way that makes it comprehensible for the strategic planner especially in health related facilities. The PREMIS method has become an established way of life in South Africa and numerous strategic condition and suitability audits have been undertaken subsequent to the large scale original National Health Facilities Audit. The processing speed of the parametric language SYMBOLIX (a recursive descent parser implemented as a stack machine) designed by the author is still very fast in comparison to what is offered by modern languages such as Visual Basic.

However the world changed significantly since the first version. PREMIS is currently being rewritten to utilise the capabilities of the Internet (PREMIS 2000i). The implicit linking technique pioneered in this system still works well (see Appendix A) except that it takes significant manpower to cross-link the alphanumeric RDBMS records with the graphic information.

The ad-hoc query builder enabled users to formulate complex (multi table joined) user defined queries to produce alphanumeric, spread sheet and graphical reports. Due to the complexity of the system very few users could unfortunately use the query builder without assistance.

The hierarchical ontology required for facilities highlighted the difficulty of creating hierarchical structures in a RDBMS.

## 4.2 The AEDES prototype system

Research undertaken over the last year at the CSIR resulted in providing a rudimentary framework for a holistic total life cycle methodology. The author wrote a prototype software system called AEDES (Architectural Evaluation and Design System). The results of this research were presented at the Eleventh Symposium on Quality Function Deployment in Detroit (Conradie & Küsel 1999).

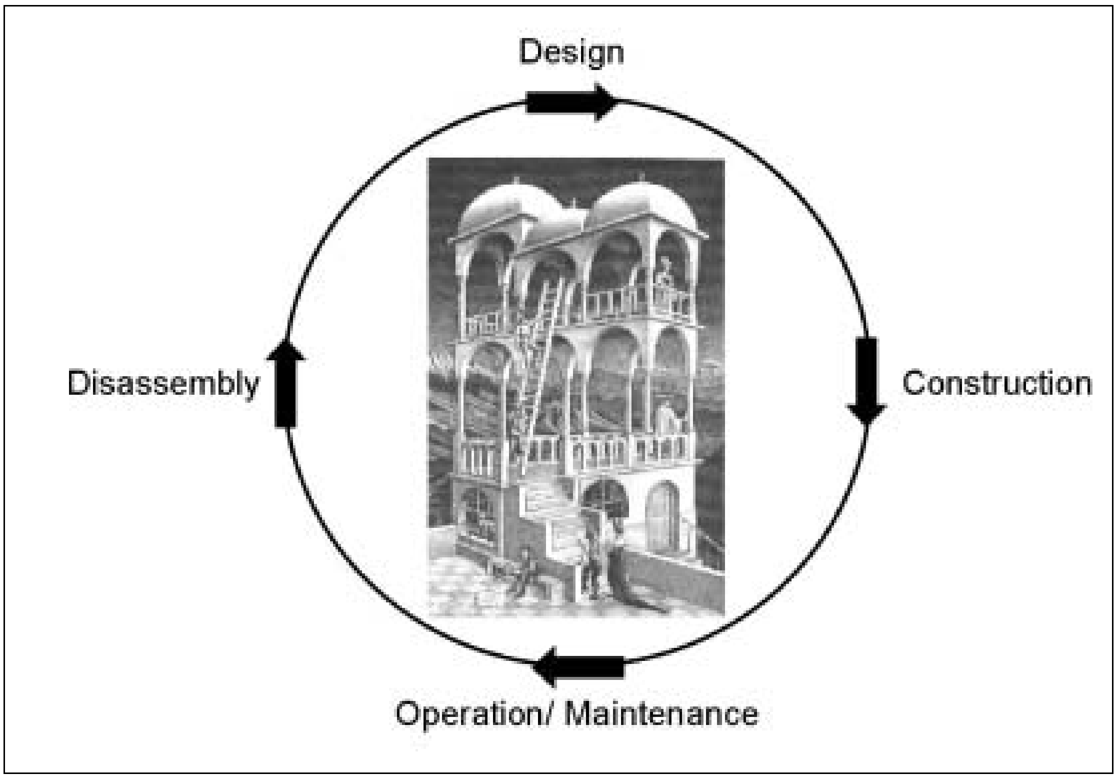


Figure 2: The life cycle phases of a building (Author)

The AEDES Prototype System offers an integrated, concurrent project environment for building designers, tailor-made for architects. It offers the following capabilities:

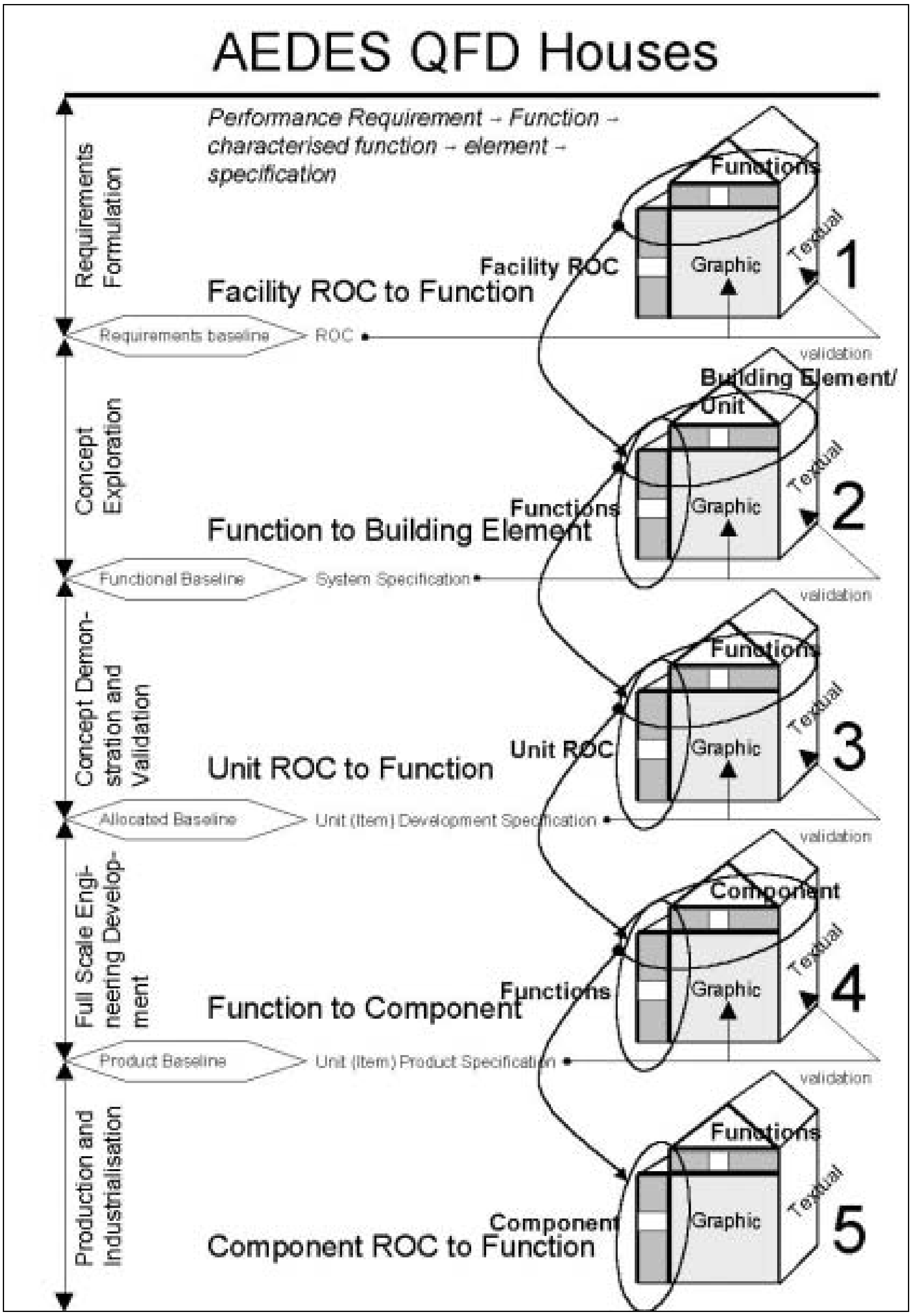


Figure 3: AEDES QFD Process (Conradie and Küsel 1999:24)

## 4.2.1 Integrated life cycle process

### 4.2.1.1 The characteristics

In AEDES a building is designed by an interdisciplinary collaborative approach, in order to derive, evolve and validate a life cycle balanced building solution. This is done to achieve optimal architecture with regards responsible, economic and quality driven design. All decisions across the life cycle of the building development process are structured, traceable and time based.

### 4.2.1.2 Evaluation during the process

AEDES supports a measuring system to facilitate structured decisions. The AEDES mission is **"To measure is to know, to quantity is to master."**

Architecture is characterised by many different fashions, trends and major movements. Various quantified indicators are required to implement a life cycle development process. For example presently there is a need to achieve sustainable development. The pillars of sustainable development are social, economic, biophysical and technical sustainability (Hill *et al.* 1998:11) To achieve sustainable development or any other technical requirement such as an energy efficient design, appropriate quantified measurements or indicators needs to be devised. These need to be quantified for the total life cycle of the building. Systems like the Building Environmental Assessment and Rating System for South Africa (BEARS) have been devised to measure certain aspects of the building operation and design (Grobler *et al.* 1997). BEARS offer a "rating" system that can be used to rate existing buildings according to a list of indicators. This list includes heating, ventilation and air-conditioning, building and furnishing materials, lighting and solar control, noise, layout, operation and maintenance issues. The South African industry offers electronic web based product databases of specific construction products. However these services do not offer any quantified material attributes that is required to base technical design analyses on.

The AEDES measuring system assists with the measurement of technical sustainability. To this end a structured generic materials library was created. A prototype materials library containing 395 materials and 760 quantified attributes were therefore constructed. Typical generic but well quantified attributes for wood are for example bending and tension parallel to grain, compression parallel to grain, compression perpendicular to grain, density, durability, modulus of elasticity and shear parallel to grain. On the basis of this the materials and components used in a proposed design solution can be technically analysed with regards factors such as life cycle cost, sustainability, condition, suitability and utilisation of the facility (Conradie 1997).

## 4.2.2 Concurrent multimedia environment

### 4.2.2.1 The need for multimedia in the architectural profession

Traditionally architects are trained to think in terms of shape, texture, colour and space. This is an analogue way of thinking, with an emphasis on the visual aspects. Therefore, design supported by multimedia would be more acceptable.

### 4.2.2.2 Multimedia in AEDES

In AEDES information is structured according to how the building will be built. This automatically ensures that design information is produced in a concurrent environment and available in the order that it is required. This information is the golden thread that enables life cycle decisions. It is envisaged that different users with multi-variant requirements will manipulate the information with software tools.

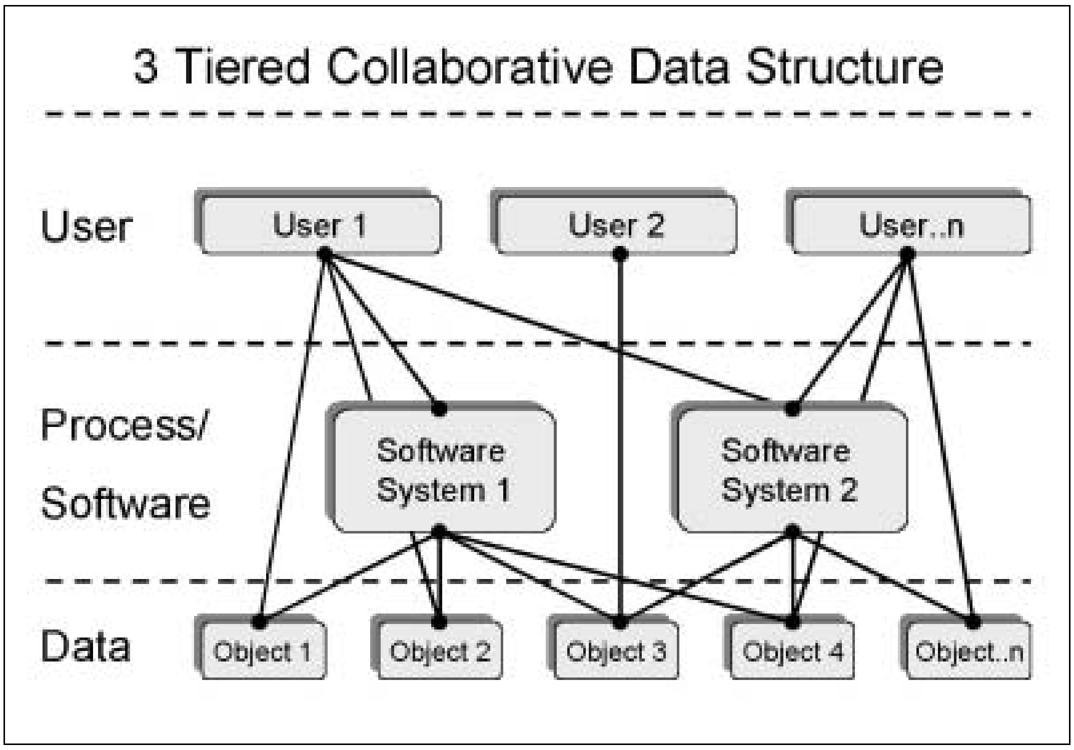


Figure 4: Three tiered collaborative data structure (Conradie and Küsel 1999:26)

Users are allocated specific information profiles to achieve certain tasks. A profile defines user rights as well as the visibility of information for that particular user and the relevant events that can occur. If the content of the task changes, then only the task profiles need to be changed without affecting the underlying information object structure. For example the maintainer responsible for replacing the light bulbs needs to access task applicable information.

The life cycle continuity of design information ensures that downstream decisions are taken within the life cycle context of the building, informed by the history of upstream actions. In this regard, it is virtually impossible to devise a suitable classification that will work across the different life cycle phases. Different life cycle phases have different information sets. For example in the operational phase, a software tool such as a Facilities Management System is used. During this phase there is a need for outsourcing, which requires an accurate specification of the task. This requires an appropriate subset of the total life cycle information set.

To accommodate different user profiles information containers were devised. All relevant project information is structured in 156 main categories implemented in a relational database.

### 4.2.3 Life cycle requirement validation

#### 4.2.3.1 Multi-media QFD

In AEDES a five-matrix House of Quality (HOQ) system is used. A System Engineering design methodology is superimposed on these matrices. This determines the structuring of data in the matrices. An architect is able to design a building starting with a raw client requirement down to component level. All design information is directly available to the development team in a concurrent multimedia project environment.

Multimedia-QFD enables the development team to validate design baselines throughout the development process (Figure 3). It offers dual validation:

> **Graphic:**
> Due to the analogue design approach of the traditional architectural design process, provision is made to capture diagrams and sketches in electronic QFD forms. When the most appropriate technical solution is sought, the standard QFD relationship matrix or context sensitive drawings, sketches and photographs back up the designer decision.
>
> **Textual:**
> Textual (alphanumeric) information facilitates the validation of design baselines and progressive design development through status reports. This implies that designer decisions are validated by substantiated documentation accumulated throughout the development process.

In AEDES the use of a generic set of client requirements (WHATs) and technical solutions (HOWs) proved to be more suitable, rather than generating a set by the Affinity Diagram method (Cohen 1995:47). 156 main categories (containers) cover performance requirements and constraints across the life cycle. Careful attention was given to ensure that data containers were correctly levelled at more or less the same level of grain. Furthermore, the system design makes it possible to theoretically create any number of matrices, vertically, as well as horizontally for any container, at any level of detail. This is perhaps a new type of matrix of matrices, albeit in a far simpler form than the Akao version (Cohen 1995:310).

When a new project is started the QFD software generates templates for WHATs and HOWs. The WHATS categories are always numbered A and the HOWS categories with a B. The HOQ1 is called the **Facility Required Operational Capability**. In this house one of the 156 containers is the Space container. This container is numbered **A1**.2.2.3 on the WHATs side and **B1**.2.2.3 on the HOWs side of the matrix. The number is indicative of the grain that originated from the data levelling exercise. The HOQ1 is rotated to the second level **Facility Specification** (HOQ2). In this process the Space container number is automatically changed to **A2**.2.2.3 on the WHATs side and **B2**.2.2.3 on the HOWs side.

The system offers a total system subsystem type analysis starting with system-level specifications down to component level of design. The following QFD system-levels are used:

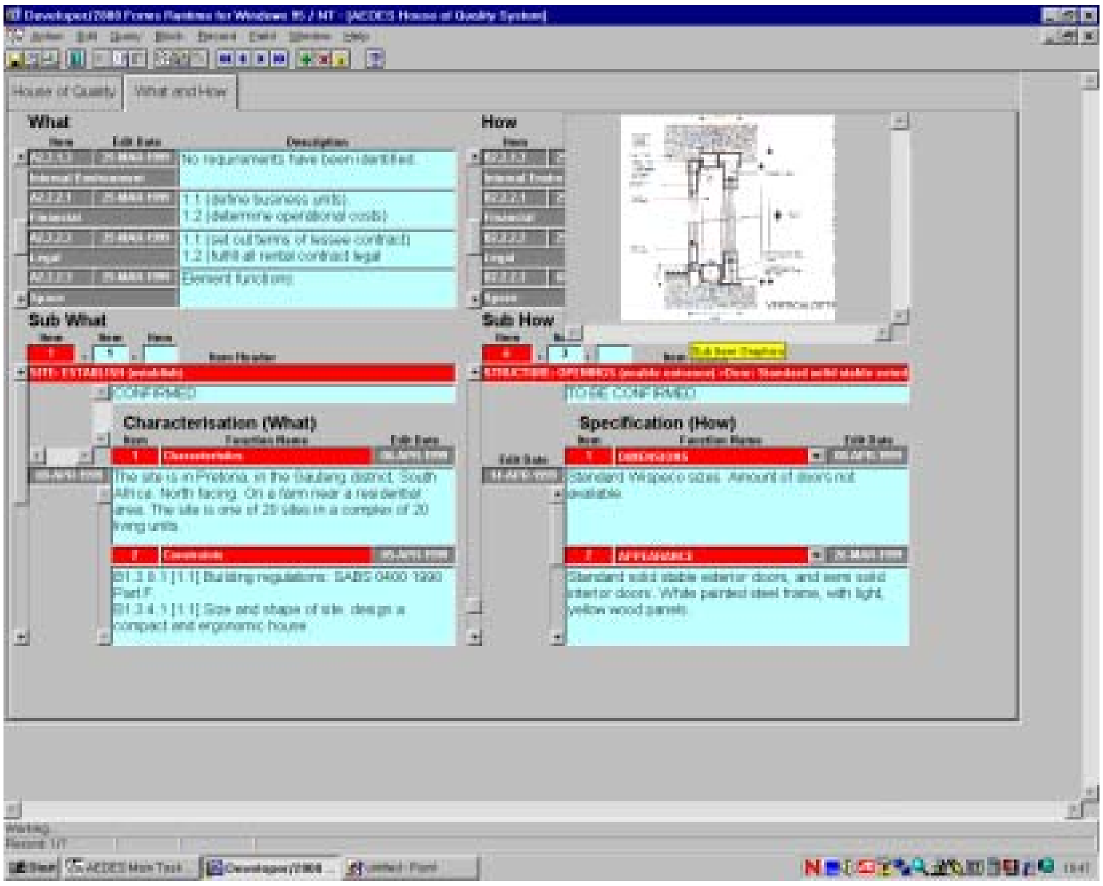
| | |
|---|---|
| House of Quality 1 | Facility Required Operational Capability |
| House of Quality 2 | Facility Specification |
| House of Quality 3 | Unit Required Operational Capability |
| House of Quality 4 | Unit Specification |
| House of Quality 5 | Component Required Operational Capability |

These five matrices handle eight classes of architectural design objects:

| | |
|---|---|
| • Complex | (Hospital complex: Group of buildings) |
| • Facility | (Hospital building and site) |
| • Department | (Administration department) |
| • Unit | (Bathroom) |
| • Zone | (Wet area) |

- Building Element (Door)
- Component (Door lock set)
- Sub-Component (Screw)

A typical AEDES computer screen (written with the ORACLE Developer 2000 Forms System) that supports multi-media is included in Figure 37.



What

| Item | Edit Date | Description |
|---|---|---|
| [illegible] | [illegible] | No requirements have been identified. |
| Internal Environment | | |
| A2.2.2.1 | 25-MAR-1999 | 1.1 (define business units) 1.2 (determine operational costs) |
| Financial | | |
| A2.2.2.3 | 25-MAR-1999 | 1.1 (set out terms of lessee contract) 1.2 (fulfil all rental contract legal |
| Legal | | |
| A2.2.2.3 | 25-MAR-1999 | Element functions |
| Space | | |

Sub What: SITE: ESTABLISH (establish)

CONFIRMED

Characterisation (What)

| Item | Function Name | Edit Date |
|---|---|---|
| 1 | Characterisation | [illegible] |

The site is in Pretoria, in the Gauteng district, South Africa. North facing. On a farm near a residential area. The site is one of 20 sites in a complex of 20 living units.

| Item | Function Name | Edit Date |
|---|---|---|
| 2 | Constraints | [illegible] |

B1.3.6.1 [1.1] Building regulations: SABS 0400 1990 Part F
B1.3.4.1 [1.1] Size and shape of site: design a compact and ergonomic house

Sub How: STRUCTURE: OPENINGS (enable entrance) >Door: Standard solid stable exter

TO BE CONFIRMED

Specification (How)

| Item | Function Name | Edit Date |
|---|---|---|
| 1 | DIMENSIONS | [illegible] |

Standard Wispeco sizes. Amount of doors not available.

| Item | Function Name | Edit Date |
|---|---|---|
| 2 | APPEARANCE | 25-MAR-1999 |

Standard solid stable exterior doors, and semi solid interior doors. White painted steel frame, with light yellow wood panels.

Figure 5: A typical AEDES screen with multi-media information (Author)

### 4.2.3.2 Break-out tools

A new concept called breakout has been pioneered. Matrices are linked to either supporting generic data forms or industry standard software analysis tools.

Generic data breakout forms enables more detailed analysis to be undertaken in order to reach a conclusion on the WHAT (characterisation) and HOW (specification) side.

A software analysis tool is linked to the QFD roof. The AEDES QFD roof is used in a slightly different way to textbook QFD. In an architectural environment it is useful to express affinity between different design objects such as units mentioned above. The relationship values have been modified to 9, 3, 0, -3, and -9. For example a value of 9 would imply that it is highly desirable to have the kitchen close to the dining room. A -9 implies that a bathroom should not be close to the living area. In complex buildings such as hospitals the correct circulation of activities and hence affinity between design units are crucial for a successful design. If the affinity decision is very complex, a standard software package such as Arena by the Systems Modelling Corporation can be used to model the proposed solution.

## 4.2.4 Ad-hoc queries and reports

### 4.2.4.1 Electronic traceability

**Structured methodology**

As discussed a System Engineering methodology is superimposed on the QFD matrices. All information accumulated during the design process is hierarchically structured and captured within a relational database (**Error! Reference source not found.**).

A building is developed following a logical sequence of steps, starting at the HOQ1 (requirement) and ending at the HOQ5 (component). The process starts with the identification of a performance requirement (For example: **SERVICES: PLUMBING**) and developed to component level. Requirements and functions fall into two main groups i.e. passive and active. Passive requirements are physical elements such as structure, services, finishes, fittings, furniture and equipment. Active requirements are activities such as enable ablutions. A set of functions required to successfully enable the operational capability of the requirement is then identified. For example a requirement for **SERVICES: PLUMBING** would have as typical functions **(supply water), (distribute water), (store water), (heat water), (control pressure), (drain waste), (drain sewerage) and (process sewerage).** These functions are individually characterised according to functional and physical characteristics and constraints. A characterised function is then allocated to a physical element, for example **SERVICES: PLUMBING (supply water) >Supply pipe.** The physical element is then specified according to 9 indicators. Firstly to physical indicators (dimensions, appearance, construction) and secondly to operational support indicators (maintenance, personnel, data, equipment, supplies and facilities).

**Starter Kits**

*"QFD is time-consuming. Worse than that, it is explicitly time-consuming, in the sense that QFD makes obvious and visible the several long meetings, attended by quite a few people. For groups that have never used QFD before, this appears as time added to their already crowded schedules. What's not as explicit or visible is the time that QFD saves." (*Cohen 1995:31).

If quality time is devoted in systematically engineering a design solution, knowledge should be saved for subsequent use. If a completed design proves useful then it can be permanently packaged as a complete generic design object. AEDES uses an object-oriented approach to this effect. In future the designer can call on these packaged objects to speed up his design process.

Over the years the Division of Building Technology of the CSIR designed and packaged architectural design starter kits, which contributed towards an accelerated and more efficient design process. The intention was to assist hospital designers with complex designs. These starter kits are available in CAD format and contain "empirical ideal" total facility layouts.

In AEDES architectural design objects are introduced in order to address the diverse and hierarchical nature of construction industry data. An object is a non-specific term synonymous with the System Engineering item for any graphic (drawing) or alphanumeric (database) data in the system. Each object belongs to a specific class and each occurrence is an instance of one of the eight object classes it belongs to. The packaged object contains 2 or 3 dimensional CAD graphic objects, full specification, characteristics and a comprehensive family of functions. The CAD object names have been carefully chosen to be compatible with the main function libraries. For example: **Class:** House and **Instance:** Residential house. It is possible to package any number of functions and main indicators for any level of architectural design object.

#### 4.2.4.2 Object manipulation

When a packaged object is subsequently retrieved, it will automatically go to the appropriate level in the QFD matrices. Each of these packaged units exists as an encapsulated world on its own. If it is brought into a specific project environment, it acquires or inherits specific localised qualities specific to the environment where it is used. At all stages full electronic traceability is maintained.

In AEDES electronic traceability can be defined as: *“The degree to which a relationship can be established between two or more products of the development process, especially products having predecessor-successor or master-subordinate relationship to one another; for example, the degree to which the requirements and design of a given system element match.”* (IEEE 1996:3). Electronic traceability ensures that the history of all actions, messages and changes are kept.

To this requirement were added event triggers that would transfer documents in the concurrent environment. All of these events are traceable as to time of occurrence and parties involved. An example is the architect that changes a layout. The AEDES system automatically generates an e-mail message to the engineer. This informs him that a change has occurred in the layout of the building and that the architect requires him to check the stability of the concrete columns.

#### 4.2.4.3 Flexible queries and reporting

The structured project data, information and knowledge facilitates flexible ad hoc queries and flexible reporting, for example:

- Report 1: Complete list of all What’s at all QFD house levels.
- Report 2: Complete list of all How’s at all QFD house levels.
- Report 3: Detailed characterisations of the solution required (QFD What side)
- Report 4: Complete specification of all items in the design (QFD How side)
- Report 5: Separate reports/customised design guides that can extract specifications for an element, according to any of the 9 indicator groups, for example a report on maintenance or equipment required.
- Query 1: Select a suitable pre-packaged unit

Report 5 is useful to assess operational requirements with regards to maintenance and to enable scientific operational comparisons to be made between different facilities of the same type.

Query 1 is used to select an appropriate pre-packaged solution. Pre-packaged units can be selected out of the data repository by any of the following means:

- Visual inspection of the characteristics.
- Structured query language based ad hoc query builders.
- Structured query language based query builders, enhanced by fuzzy set logic.
- Electronic Pugh concept selection.

Data repositories could be web-based and be provided by third party developers in a similar way that building product information is acquired. When a selection is made, the object is brought into the project environment and localised with regards the local material prices, maintainers and equipment required.

#### 4.2.4.4 Implicit linking technique

It is a technique originally pioneered in the development of the CSIR Facilities Management system, PREMIS developed by the author. This technique creates relationships between diverse sources of alphanumeric and graphic data. This facilitates connection between existing commercially available information and new alphanumeric and graphical objects created (Conradie 1996). Appendix A describes this method in detail.

### 4.2.5 Major components

#### 4.2.5.1 Relational database

In AEDES data relate to the life cycle of a building and is continuous and concurrent. They are accessible by appropriate software tools. All software components are well integrated with the underlying stratum of data, by means of specific AEDES diagramming and data manipulation tools.

AEDES incorporates a relational database management system, based on prior experience. It supports world-wide-web deployment. It is in the nature of structured methodologies that a significant amount of data is generated. The database structure is such that it is possible to cut the data horizontally or vertically. It is possible to extract all specifications for all indicator categories or only construction related items. If a specific data object is changed all dependant documentation will be synchronised. If a design is changed the impact with regards the main indicator categories can be assessed easily.

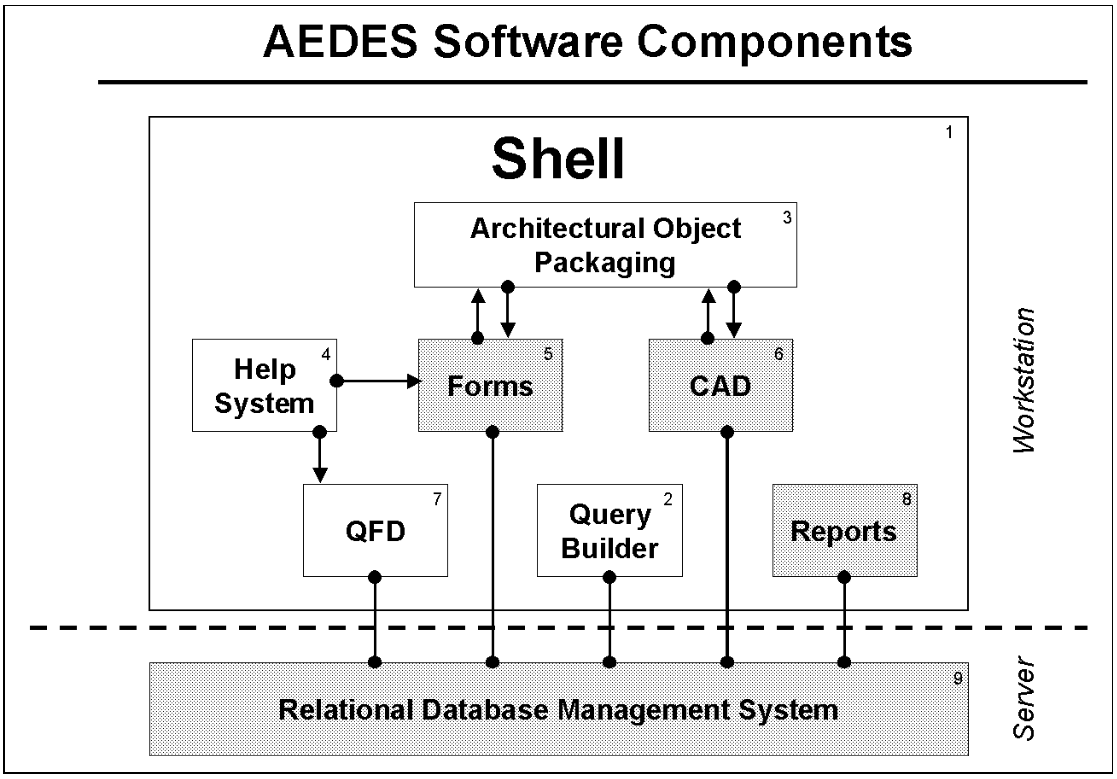

AEDES Software Components
Shell
1
Architectural Object Packaging
3
Help System
4
Forms
5
CAD
6
QFD
7
Query Builder
2
Reports
8
Relational Database Management System
9
Workstation
Server

Figure 6: AEDES software components (Author)

### 4.2.5.2 Software shell

The software shell provides an integrated environment for the development team. Users identify themselves by means of user names and passwords giving them certain access rights and user profiles. The ad hoc query builders as well as the standard reports are accessed from the shell.

### 4.2.5.3 Database forms

The database forms are designed in such a way as to give the user visual clues as to what is required. The intention is that the main QFD data capture forms will be used in conjunction with the help system. Users may cut and paste template examples from the help system to assist them to quickly populate the required form data fields.

The AEDES forms are completely generic. Any complexity of architectural design object can be conveniently handled. Any level can contain any number of sub-objects. The only restriction is that the description per text object or sub-text-object may not be more than 2 000 characters. The size of any graphic object may not be more than 2 Giga bytes. For practical purposes the system has virtually no limit.

### 4.2.5.4 Help system

A comprehensive context sensitive help system is provided, due to the fact that architectural practitioners are new to the field of structured briefing design. The help file provides a step by step description of the various actions that need to be taken in the briefing and design process. New definitions were invented for the intended environment to make it more acceptable.

### 4.2.5.5 QFD diagram software

Visio software proved useful to generate large pre-printed QFD charts for QFD sessions. Visio is electronic drawing sheet based with sufficient programming support to connect it to the central server-based database or even a portable personal database. The intention is that a scribe, familiar with the software, will be seated out of the way of the group's activities and quietly record data into the computer as the team reaches consensus. The decisions can then be copied onto large wall charts or even re-plotted from Visio. Decisions are captured in the main Oracle database forms that ensure full traceability and integrity. The Visio diagrams are used to capture the data required for matrices such as the relationships, technical matrix, planning matrix and technical correlation. Relationships can be defined across the container categories such as **A1.3.4.1 Landscape features** and **B1.2.2.3 Space**. At a finer grain items within the space container can be related such as a relationship between **A2.2.2.3**.*2.3* **Structure**: *Superstructure* and **B2.2.2.3**.*2.5* **Structure**: *Roofing*. At this stage the authors are of the opinion that due to the fact that the design is related to an architectural domain, the data will tend to cluster around physical containers and especially the space category **A1.2.2.3** through to **A4.2.2.3**

### 4.2.5.6 Starter kit packaging

A first attempt, albeit in very raw form, was made to package architectural design knowledge. The starter database kit form attempts to enable a designer to package an architectural object in a 2 or 3D CAD drawing, inclusive of main characteristics, specification and related materials. No attention was given to the tacit knowledge aspects or the intelligent interoperability of objects.

#### 4.2.5.7 Materials database

The materials database form makes provision for the main material description, a generic set of user definable attributes and a photograph or technical diagram depending on the type of material.

### 4.2.6 Conclusion

The AEDES Prototype System provides a new prototype structured design methodology, within an integrated concurrent multimedia project environment. An attempt is made to turn certain parts of the architectural briefing and design process into a science without compromising architecture or existing design ethics. According to Kolodner[1] the intention should not be to turn Architecture into a science but rather to create good tools that maximise the efforts of creative designers. This ensures the highest possible level of competitiveness, professionalism and use of scarce resources for architectural practitioners in a developing country.

In order to bring AEDES closer to commercial realisation and to protect the existing substantial investment in the FM system PREMIS (Professional Real Estate Management Information System) more focussed research is required. The two most pressing needs in South Africa are:

- The creation of a World Wide Web enabled continuous data infrastructure over the life cycle of a building.
- The ability to store and retrieve carefully engineered design knowledge in the form of electronic design starter kits.

In the present AEDES prototype system the abovementioned points are still poorly researched although the theoretical need can be clearly identified.

### 4.2.7 Critique of AEDES

AEDES was a first attempt to create an Integrated Project Environment that could span the entire life cycle of the facility. The system managed to use aspects of QFD, Systems Engineering and Concurrent Engineering. The research and development team (Conradie and Küsel 1999) succeeded in creating a system that enables the designer to start with a raw client requirement and go right down to the component level. Functional decomposition was used extensively.

The most significant shortcomings of the system were:

- An over emphasis of the functional decomposition (transformational approach) aspects led to a rigid system that is very prescriptive to the creative people that is supposed to use it.
- It was difficult to map multiple functions from different main container categories identified in AEDES to physical structural elements.
- The team never fully achieved the objective of creating starter kits[2] (cases at various levels of specificity) to expedite future designs. This was due to the fact that the benefits that AI could offer and specifically CBR was not known at the time.

---

[1] Kolodner, J., Georgia Insitute of Technology, 2000 – personal communication

[2] A starter kit is a simple architectural CAD drawing that contains a complete ideal layout of for example a hospital ward and could be used to rapidly design a hospital. This is useful in South Africa with the shortage of skilled designers. The division of Building Technology of the CSIR developed an extensive set of these designs that are currently used to design significant hospitals and clinics. See 4.2.4.1 Starter Kits.

- Adaptation of designs was very difficult due to the rigid structure.

AEDES highlighted the enormous complexity of creative design in an open world that is confirmed by numerous authors. The AEDES team failed to understand the implications of the fact that *design constraints are not constructive* and that the *design problem spaces are not enumerable* (Hinrichs 1991:15-16).

## Summary

The long operational life of PREMIS can be attributed to the fact that it is modular (primitive object oriented structure) but also succeeded in separating data, process and application clearly. These facts made the system flexible. This provided useful insights into the design of applications that use the same data infrastructure yet serve totally different clients. The implicit linking technique is a useful way to link object oriented graphic and alphanumeric information together.

PREMIS provided useful insights into a deeper understanding of ontology in a Facilities Management environment. The development also highlighted the inadequacies of Relational Databases with regard hierarchical structures. Facility Managers are often confronted with the problem of maintaining construction structures that were badly designed. This emphasises the importance of the early phases of the building life cycle.

Although AEDES can be viewed as a failure in commercial terms it provided the first insights into the how user requirements might be extracted and structured to obtain a performance requirement. It was discovered that requirements and functions fall into two main groups i.e. active and passive. A set of functions to successfully enable the operational capability of the requirement could be identified. These functions could be individually characterised according to functional and physical characteristics and constraints. Finally a characterised function could be allocated to a physical element.

# Chapter 5: Aims of ARGOS

## Introduction

Chapter 3 and 4 indicate that an opportunity exists to bridge the explicit and tacit aspects of design with an intelligent component. This chapter introduces the main aims of ARGOS and also discusses the important technique of concept selection (Pugh 1996; Ulrich *et al*. 1995:105-122). Concept selection takes place in the phases of design. The examples are used to develop possible conceptual solutions on the basis of the experience gained with the precedent systems discussed in Chapter 5 whilst at the same time describing the process. The concepts are specifically hand drawn to emphasise the creative and exploratory nature of what concept selection should be. On the basis of this a detailed prototype implementation for ARGOS will be developed in Chapter 6.

From the previous chapters it is clear that it is very difficult to improve the briefing and design processes successfully. Systems that attempt to improve these processes need to be very flexible. Designers work in many different equally valid ways. The design of the ARGOS intelligent component should be such that it can manipulate design information easily. Hinrichs (1991) and Simina (1999) provide useful insights as to the direction that should be taken specifically with regards creative design. It is clear that the creative human designer should remain in control and that systems should be designed in such a way as to assist the human designer. The following significant areas of assistance can be identified that could possibly be improved with ARGOS on the desktop:

- The ability to select the best concept for a project or part of project
- The availability of previous design experience at various levels of specificity to remind the designer of aspects he might have forgotten
- The ability to test shape and placement of design parts (relationships)
- To ability to judge the suitability of a design with regards function and performance (scenario planning)
- Long life and persistence of design fragments in a neutral environment
- Collaboration in a multi-disciplinary team on a global basis
- Support for modelling and simulation.

New branches of Artificial Intelligence (AI) such as Case-based Reasoning (CBR) brought realism as to the possible contribution AI could make in this environment. Current AI attempts to assist the human designer, not to simulate the capabilities or improve the human brain. To this end CBR has already contributed significantly to capture experiential knowledge. The study and also practical experience indicate that it is unlikely that all design knowledge could ever be concentrated in a single location or database. In the present world knowledge is added at such a tempo in general that attempts to gather it in a single database is unlikely to succeed, because of the diverse formats that exist in the world of design. Even after 20 years the world of CAD has not succeeded in formulating data exchange standards that can connect these diverse systems reliably beyond the pure exchange of graphical information. It must be admitted that although many different standards have seen the light, they only satisfy very specific needs. Due to the different ontological needs of design knowledge over the life cycle data standards such as the Industry Foundation Classes (IAI), IGES and DXF are not really suitable if CBR needs to be introduced in a neutral environment.

The World Wide Web has grown beyond recognition as the largest network the world has ever seen. Cheswick and Burch at Bell Laboratories  (2 000) recently mapped the Internet by means of electronic tracers or packets. In the process at least 88 000 main routers were discovered. These routers are connected to millions of individual users. It became clear that an intelligent design component should both be object based and Internet enabled. Its knowledge must be structured and self-documenting. It should be able to operate in a wide variety of environments over a long period of time. It must be useful in small and large project teams using different design and construction processes. It cannot be predicted with certainty what the nature of these processes will be in future.

Due to these diverse requirements the author came to the conclusion that the optimal solution would be the introduction of an intelligent design components. These must be highly flexible and be able to operate in very diverse desktop container environments without being dependant on single technologies such as databases, CAD or other propriety programs for its success. Riesbeck (1996) stated that his theme for *post-modern* AI is the concept of intelligent components. The goal should be the improvement of how systems function through the development of intelligent parts to those systems. In *post-modern* AI, AI becomes an invisible part of the overall system. The goal is not smart appliances and cars that talk to the user. The goal is now street lamps that do not waste electricity on totally deserted sidewalks and traffic lights that do not turn green for streets closed for construction. Riesbeck (1996) indicates several research areas for CBR relevant to making CBR feasible for intelligent components in non-intelligent systems such as the indexing and adaptation aspects.

Concept selection is discussed under 5.1 for the following reasons:

- Concept selection is one of the most critical and difficult problems in design.
- The examples are used to develop possible conceptual solutions on the basis of the experience gained from Chapter 5 for the detailed implementation of ARGOS in Chapter 6 whilst at the same time describing the process.

## 5.1 Concept selection

### 5.1.1 Introduction

Concept selection is the emergence and selection of the best and strongest concepts with respect to customer needs and other criteria. Although creativity is essential throughout the entire product development process, concept selection reduces the number of alternatives under consideration. Concept selection is one of the most critical and difficult problems in design. It is the selection of the optimal concept with which to proceed to detail and ultimately manufacture or in the case of architecture construction. A lack of thoroughness in concept selection will bring conceptual vulnerability. Due to the nature of architecture it is easy to select the wrong concept and difficult to select the optimal one. In practice it is impossible to evaluate all possible solutions to a particular problem.

Concept selection is an integral part of the product development process. By using different methods the design team generates alternative concept solutions. According to Ulrich (1995:111) the use of a structured concept selection method offers the following main benefits:

- *A customer focused product*. Concepts are evaluated against customer-oriented criteria.
- *A competitive design*. The benchmarking of concepts with respect to existing designs push the designers to exceed their competitors' performance along key dimensions.
- *Better product-process co-ordination*. Explicit evaluation of the product with respect to manufacturing criteria improves the product's manufacturability and helps to match the product with the process capabilities of the firm.
- *Reduced time to product introduction*. A structured methodology becomes a common language among the design team members such as manufacturing and industrial engineers. This avoids ambiguity, improves communication and reduces false starts.

- *Effective group decision-making.* A structured methodology encourages decision-making based on objective criteria and minimises the likelihood that arbitrary or personal factors influence the product concept.
- *Documentation of the decision process.* A structured method results in a documented rationale behind concept decisions. This record is useful for integrating new team members. It is also useful to assess the impact of changes in the customer needs or in the available alternatives.
- *Traceability.* The commitment or "buy-in" of team members is recorded.
- *Structured decision.* Should it be necessary to backtrack on decisions it can be done in a structured way.

The following disadvantages can be identified:

- Concept selection could force the team to stick to the beaten track.
- It could inhibit lateral thinking.
- "Hard issues" and "hard people" could dominate the decisions taken.
- It is more work to analyse design problems in this structured way.

### 5.1.2 Conceptual vulnerability

Pugh (1996:169) states that conceptual weakness in design manifests itself in two ways:

- The final concept is weak due to lack of thoroughness in the conceptual approach. Thereafter no amount of attention to technical and detail requirements will save the situation.
- The final concept is good and the best within the constraints. Due to lack of thoroughness in conceptual approach and selection the reasons for its strength are not known or fully understood. It is also difficult to persuade or refute others on the basis of a sound technical argument.

### 5.1.3 Overview of the method

Concept selection is often performed in two stages to manage the complexity of evaluating dozens of product concepts. Screening is a quick evaluation aimed at producing a few viable alternatives. Scoring is a careful analysis of these concepts to choose a single concept that will lead to product success. Concept screening follows a seven-point process and scoring a six-point process that leads the team through the activity. If the design decisions are simple then concept screening is adequate. As an example four hand drawn alternative concepts are included in Figure 39 to Figure 42. This illustrates the design of a user interface for starter kits (intelligent case enabled components) that will be discussed in much more depth in chapter 6. The reference concept is the type of CAD drawings drawn in MicroGDS that is currently being made available on the Internet to architects designing hospitals (Figure 43). At the moment CAD is a mature technology and is widely used by most design professionals.

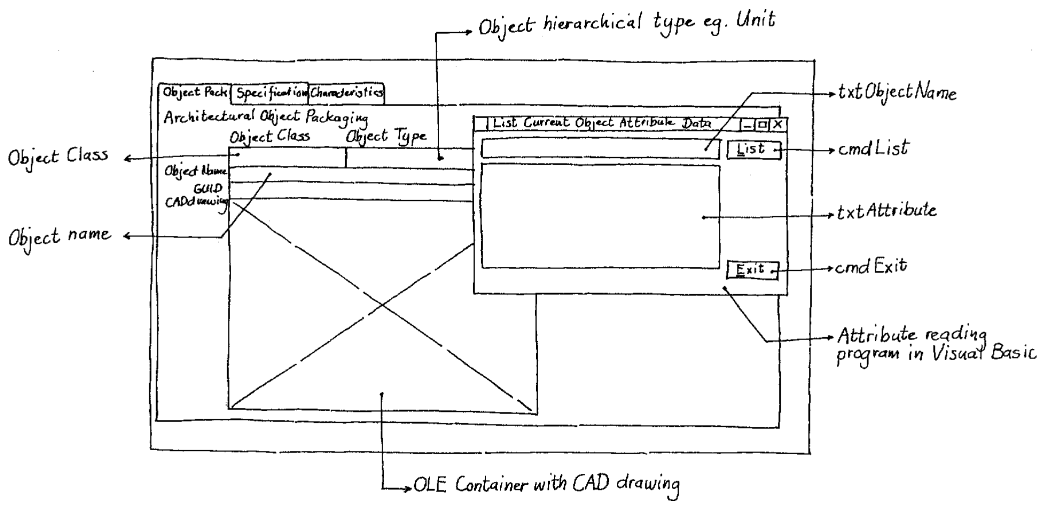

Object hierarchical type eg. Unit
Object Pack
Specification
Characteristics
Architectural Object Packaging
Object Class
Object Type
Object Name
GUID
CADdrawing
List Current Object Attribute Data
List
Exit
Object Class
Object name
txtObjectName
cmdList
txtAttribute
cmdExit
Attribute reading program in Visual Basic
OLE Container with CAD drawing

Figure 1: Concept A, Oracle form with CAD in OLE container and Visual Basic attribute reader (Author)

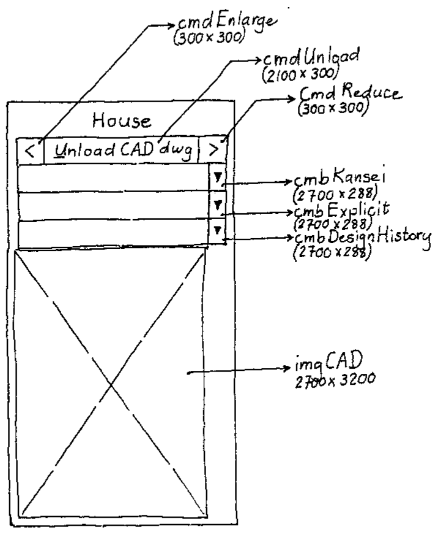


Figure 2: Concept B, ActiveX control based starter kit (Author)

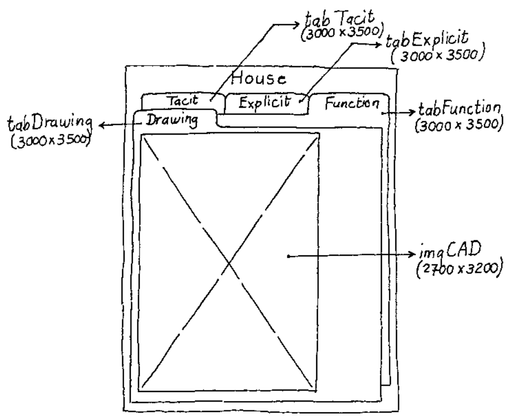


Figure 3: Concept C, ActiveX control based starter kit (Author)

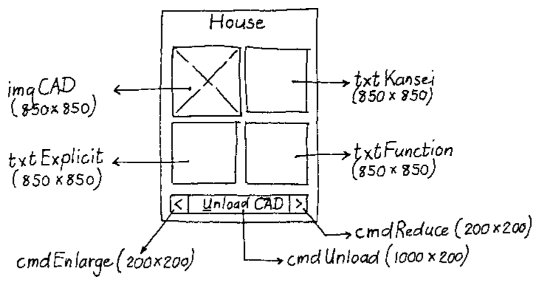


Figure 4: Concept D, ActiveX control based starter kit (Author)

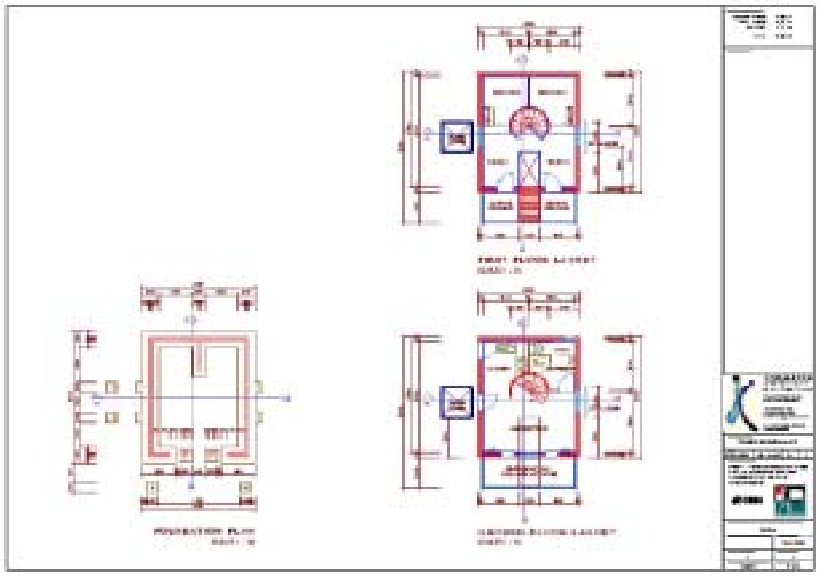


Figure 5: Typical starter kit drawing as used in the design of the AEDES prototype used as the starter kit reference concept (Author)

The following procedure could be used for concept screening:

1. Select possible solutions to the particular design problem.
2. Prepare the evaluation matrix.
3. Rate the concepts.
4. Rank the concepts.
5. Combine and improve the concepts.
6. Select one or more concepts.
7. Reflect on the results and the process.

If a *Case-Based Reasoning* (CBR) methodology were to be applied, then point 1 would be equivalent to a *case retrieval* process (Kolodner 1993:17). Concept scoring is identical to screening except that point 1 is not applicable.

### 5.1.4 Concept screening

#### 5.1.4.1 Select possible solutions to the particular design problem

It is important that semantics are clarified and that the team members attach the same meaning to the criteria. The design team gathers all possible potential solutions for the design problem. Sketches are produced to the same level of detail in each case if it is a manual system.

#### 5.1.4.2 Prepare the evaluation matrix

The selected concepts are entered on the matrix. The concepts are best displayed with a written as well as a graphic representation. If the team is considering more than 12 concepts, then voting should reduce them. The selection criteria are listed on the left-hand side of the screening matrix. The needs should be based on the needs of the customer as well as of the enterprise. At this stage the criteria should be at a reasonably high abstraction level and should contain about 5 to 10 metrics or criteria. It is important that the criteria should be selected in such a way that a distinction can be drawn between the different concepts. The team should also avoid to list too many unimportant criteria because during the screening phase each criterion is weighted equally. The criteria must be on the same basis and all to the same generic level.

Examples of initial screening criteria in an architectural environment could be *area and volume measurement, comfort indicators, acoustical qualities, physical shape and location, materials, energy and abstracted design metrics* (Conradie 1997). Due to the general complexity of criteria in an architectural environment the Saaty (1980) analytic hierarchy process might be more appropriate for comparing and evaluating different design solutions on a technical basis. Saaty (1980) describes advanced examples such as conflict analysis for health care management, energy examples such as optimum choice of coal plants and energy storage systems.

After careful consideration the team chooses the reference concept against which all the other concepts will be rated. If design or industry standards already exist for the product under consideration this should be included in the matrix to form a datum choice. The datum could be any of the following:

- A commercially available product.
- An earlier version of the product.
- Any of the concepts under consideration.
- Combination of subsystems combined to represent the best features of different products.

#### 5.1.4.3 Rate the concepts

During this stage each concept is a general notion of the final product. Weighting of the ratings or detail is therefore not relevant at this stage. The spreadsheet was implenented in Excel and the command button used VBA to calculate the results (Figure 44).

Each concept’s criterium is rated against the chosen datum by means of the following legend:

- A plus sign (+) means *better than*, *less than*, *less prone to*, *easier than* relative to the datum.
- A minus sign (-) means *worse than*, *more expensive than*, *more difficult to develop than*, *more complex than*, *more prone to*, *harder than* relative to the datum.

- Where any doubt exists as to whether a concept is better or worse than the datum, use a (0) that means the same as the datum.

If objective metrics are available they should be used. In the design of a shopping centre *usable* and *rentable* area could give a very good indication of the commercial viability of a design. These metrics minimise the judgmental error that is inherent in the process. Objective metrics suitable for concept screening can also be derived from the establishment of target specifications of the product. At this stage it can also be established whether some criteria need further investigation.

#### 5.1.4.4 Rank the concepts

After rating the concepts the team sums the ratings. The sum of +’s, -’s and 0’s are first derived. The net score is then obtained (Figure 44). Once the summation has been calculated the concepts can be ranked. It is convenient at this stage to identify the differentiating criteria, which really make the biggest difference.

#### 5.1.4.5 Combine and improve the concepts

After the concepts have been ranked the results should be verified to make sure that they make sense. After assessment of the individual scores the following possible phenomena could be observed:

- A certain concept exhibits exceptional strength. In this case the matrix should be rerun with the strengths removed. If, as a result of running the matrix several times, the initial high scores persist they are likely to be the best concepts with which to proceed.
- A strong pattern of concepts does not emerge and each concept appears to have a uniform strength. This is very unusual. In this case the datum should be changed and the pattern reassessed.
- A particular concept persists. The datum should be changed and the process repeated. If the result remains the same the emergent strong concept can assume the role of datum. The matrix should then be rerun and the results assessed.

Ways can now be considered to combine and improve certain concepts. Two possibilities are:

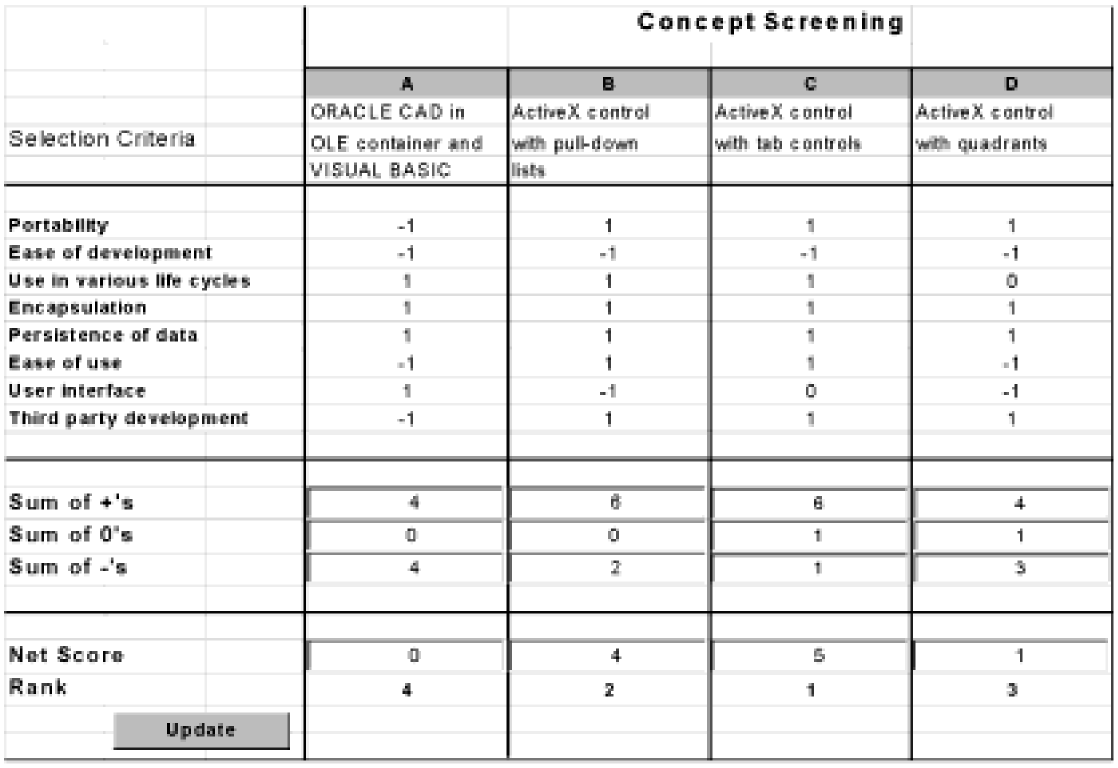
- Is there a good concept that is degraded by one single bad feature? Can a small modification improve the overall concept and maintain an edge over the other concepts?
- Can concepts be combined in such a way as to improve the good characteristics?

#### 5.1.4.6 Select one or more concepts

At this stage it is clear what each concept is worth and a decision can be taken which concepts can be further analysed and refined. It must also be decided whether another round of concept screening will be performed or whether the team will proceed to concept scoring. If the screening did not provide sufficient clarity then the more detailed concept scoring can be used.

#### 5.1.4.7 Reflect on the results and the process

All the team members should agree with the outcome. If somebody does not agree with the results it is possible that crucial criteria are missing from the matrix or a particular rating could be wrong. It is important that the team agrees with the results, because it increases the commitment to subsequent product development stages.



| | Concept Screening | | | |
|---|---|---|---|---|
| | A | B | C | D |
| Selection Criteria | ORACLE CAD in OLE container and VISUAL BASIC | ActiveX control with pull-down lists | ActiveX control with tab controls | ActiveX control with quadrants |
| **Portability** | -1 | 1 | 1 | 1 |
| **Ease of development** | -1 | -1 | -1 | -1 |
| **Use in various life cycles** | 1 | 1 | 1 | 0 |
| **Encapsulation** | 1 | 1 | 1 | 1 |
| **Persistence of data** | 1 | 1 | 1 | 1 |
| **Ease of use** | -1 | 1 | 1 | -1 |
| **User interface** | 1 | -1 | 0 | -1 |
| **Third party development** | -1 | 1 | 1 | 1 |
| **Sum of +'s** | 4 | 6 | 6 | 4 |
| **Sum of 0's** | 0 | 0 | 1 | 1 |
| **Sum of -'s** | 4 | 2 | 1 | 3 |
| **Net Score** | 0 | 4 | 5 | 1 |
| **Rank** | 4 | 2 | 1 | 3 |

Update

Figure 6: The concept screening matrix for the concepts A to D (Author, based on Pugh 1996; Ulrich et al. 1995:114)

## 5.1.5 Concept scoring

Concept scoring is used when the previous concept screening could not provide sufficient differentiation. During scoring more detail is considered and the relative importance of the selection criteria is considered. The concept scores are determined by the sum of the different rating weights. In this case a six-stage process is followed. Point 1. Under concept screening is omitted this time.

The designers proceed to develop the strongest concepts emerging from the initial evaluation. The concepts are now engineered to a higher level of detail. The additional work results in a greater understanding of the problem and its solutions. Such understanding leads to a refinement and expansion of the criteria for evaluation. The matrix is reformed to incorporate the enhanced concepts and also the revised or expanded criteria. The general mechanism of the first phase is repeated. The outcome will confirm the earlier patterns or give rise to a reordered set of concepts. In each case the designer should have a critical review of the emergent pattern.

### 5.1.5.1 Prepare the selection matrix

The process is the same as previously with the screening stage. Due to the more complex calculations a computer spreadsheet is a convenient way to do calculate the results. The concepts are now more detailed than during the screening phase and therefore more detail is available. The concepts that have been selected are entered on the top of the matrix. More detail can be added to the selection criteria. A hierarchical approach can be followed where general terms such as “comfortable livingroom” could be broken down into “Internal

Temperature”, “Lighting Levels” and “Ventilation”. The level of detail depends on the needs of the designer.

After the criteria have been entered, the team adds importance ratings to the matrix. Several different schemes can be used to weigh the criteria, such as a scale of 5, 4, 3, 2, 1 or a distribution of 100 percentage points amongst the criteria. Refined marketing techniques exist that can be used to determine weights from customer data.

**5.1.5.2 Rate the concepts**

As in the screening stage the concepts are compared to the reference concept. However at this stage a finer scale is used to give finer resolution. A 1 to 5 five scale is normally used. It is generally best to have a scale that is symmetric around the reference concept score.

| **Relative Performance** | **Rating** |
|---|---|
| Much worse than reference concept | 1 |
| Worse than reference concept | 2 |
| Same as reference concept | 3 |
| Better than reference concept | 4 |
| Much better than reference concept | 5 |

Unless by coincidence the reference concept is of average performance with regards all criteria, the use of the reference concept for the evaluation of all criteria will lead to scale compression for some of the criteria. For example if the reference concept happens to be the easiest construction, then all remaining concepts will get 3, 2 or 1 scores. In a situation like this the team can choose different concepts as the reference points for different criteria.

**5.1.5.3 Rank the concepts**

Once the ratings are entered for each concept, weighted scores are calculated by multiplying the raw scores by the criteria weights. The total score for each concept is the sum of the different scores. Each concept is given a rank according to its total score. The following formula is used:

$$s_j = \sum_{i=1}^{n} r_{ij} w_i$$

where $r_{ij}$ = raw rating of concept *j* for the *i* th criterion

$w_i$ = weighting for the *i* th criterion

$n$ = number of criteria

$s_j$ = total score for concept *j*

**5.1.5.4 Combine and improve the concepts**

As in the screening stage, the team looks for changes that would improve the concepts reviewed. The team is now aware of the strengths and weaknesses of certain features of the product concepts.

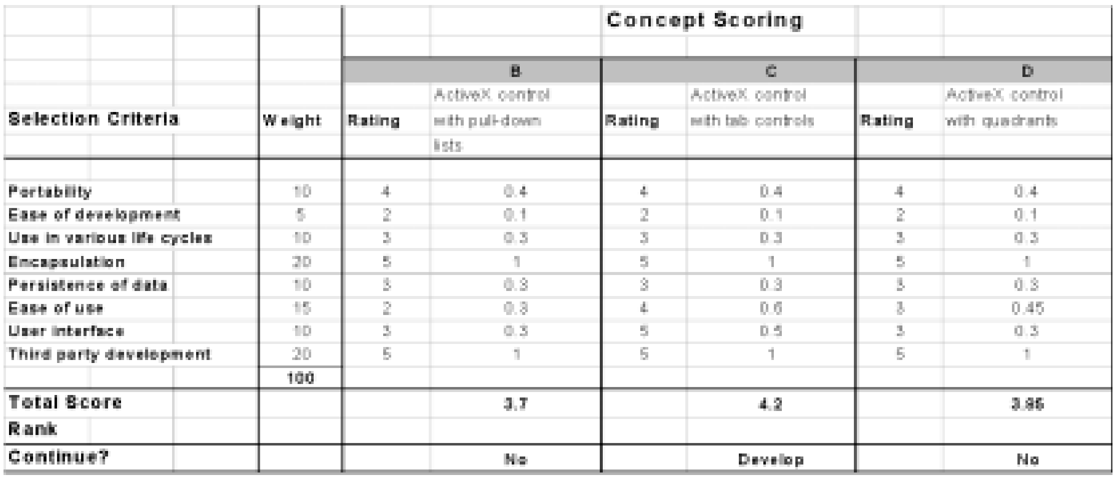
### 5.1.5.5 Select one or more concepts

The final selection is not just to select the concept that achieves the highest ranking. The team should explore its initial evaluation by conducting a sensitivity analysis. As is the case in Figure 44 a spreadsheet can be used to vary weights and ratings to determine their effect on the ranking (Figure 45).

By careful analysis of how sensitive a ranking is to variation in a particular rating, it can be determined whether uncertainty about a particular rating has a large impact on the choice. Sometimes it is advisable to select a lower scoring concept that has less uncertainty than the high scoring one.

If the customer groups have different preferences then two different matrices can be prepared. It is possible that one concept is dominant in both cases. In the examples given it is clear from the data that Concept C is the most promising.

### 5.1.5.6 Reflect on the results and the process



| | | Concept Scoring | | | | |
|---|---|---|---|---|---|---|
| | | B | | C | | D |
| Selection Criteria | Weight | Rating | ActiveX control with pull-down lists | Rating | ActiveX control with tab controls | Rating | ActiveX control with quadrants |
| Portability | 10 | 4 | 0.4 | 4 | 0.4 | 4 | 0.4 |
| Ease of development | 5 | 2 | 0.1 | 2 | 0.1 | 2 | 0.1 |
| Use in various life cycles | 10 | 3 | 0.3 | 3 | 0.3 | 3 | 0.3 |
| Encapsulation | 20 | 5 | 1 | 5 | 1 | 5 | 1 |
| Persistence of data | 10 | 3 | 0.3 | 3 | 0.3 | 3 | 0.3 |
| Ease of use | 15 | 2 | 0.3 | 4 | 0.6 | 3 | 0.45 |
| User interface | 10 | 3 | 0.3 | 5 | 0.5 | 3 | 0.3 |
| Third party development | 20 | 5 | 1 | 5 | 1 | 5 | 1 |
| | 100 | | | | | | |
| Total Score | | | 3.7 | | 4.2 | | 3.95 |
| Rank | | | | | | | |
| Continue? | | | No | | Develop | | No |

Figure 7: The concept scoring matrix for the concepts B, C and D (Author, based on Pugh 1996; Ulrich et al. 1995:117)

Finally the team reflects on the selected concept and on the concept selection process. This is an important point where the entire team should be convinced that all the relevant issues have been discussed and that the selected concept will satisfy customers and will be economically viable.

### 5.1.5.7 Some important factors

If the technique proposed is to be used in architectural design then the following points need to be considered carefully.

- *Decomposition of concept*. The idea of concept selection is that the selection criteria and therefore customer needs can be evaluated separately. It is also assumed that concept quality is the sum of the quality of each criterion. It is difficult to decompose certain products into a set of independent criteria. This is often the case in architecture due to the complex relationships among design criteria. Architectural design is a good example of multi-attribute decision making. Many of the relationships are non-linear. By means of QFD the customer preferences and the relative importance attached to certain criteria can be clearly established. This should be used in the concept selection process.

- *Subjective criteria.* Some selection criteria especially the ones related to aesthetics are highly subjective. In this case the team cannot decide on those issues on behalf of the customer. It is recommended that an accurate subjective voice of the customer be obtained by means of techniques such as Kansei Engineering discussed in chapter 3.

- *Where to include cost.* The selection criteria are mostly customer needs that could have originated from the use QFD techniques as suggested in the AEDES prototype. "Ease of manufacturing" and "manufacturing cost" are not customer needs. Depending on the type of construction project cost is an important factor in choosing a concept. To facilitate concept selection some measure of cost and life cycle cost should be included into the evaluation matrix.

- *Elements of complex concepts.* Some complex concepts are aggregations of simpler concepts. If all the concepts are aggregations of several simpler concepts, then the simple concepts can first be evaluated independently.

- *Ubiquitous use of concept selection.* Concept selection should be used throughout the product development process and not just in the beginning. The same is true of the numerous other techniques available such as QFD, Kansei, Theory of Constraints and TRIZ.

### 5.1.6 Enhanced QFD and concept selection

When QFD was first introduced into the U.S.A., the QFD model assumed that the selection of appropriate technology for a product was outside the scope of QFD (Cohen 1995:182). Don Clausing and Stuart Pugh realised that the process for selecting innovative concepts should interact with the translation of customer needs to prioritised technical responses. These ideas were embodied in a process called Enhanced QFD (EQFD). EQFD consists of five interrelated processes:

- Contextual analysis and static/ dynamic analysis.
- Structuring of product design specification.
- House of Quality.
- Concept selection.
- Total system/ subsystem analysis.

The concept selection process as described by Pugh is pivotal in EQFD. It is recommended that the EQFD be facilitated and managed by a person that is not directly involved in producing concepts to avoid suspicion of favouring a particular idea or interest.

## Summary

An intelligent design component should be both object based and Internet enabled. Its knowledge should be structured and self-documenting. It must be able to operate in a wide variety of environments over a long period of time. It should serve small and large project teams using different design and construction processes. It cannot be predicted with certainty what the nature of these processes will be in future.

This chapter identified the significant areas of assistance that an autonomous intelligent component such as ARGOS could improve such as:

- Concept selection

- Retrieval of design experience
- Test of relationships
- Scenario planning
- Collaboration on a global basis
- Modelling and simulation

# Chapter 6: Implementation details

## Introduction

This chapter discusses the main building blocks that are necessary to implement ARGOS such as the life cycle information infrastructure, constraints and the design of a conceptual ARGOS. It explores the role that ARGOS could play in structured planning and design knowledge delivery. The relationships between the ARGOS, ActiveX design object and typical applications software are explored.

The basis for successful implementation is the formulation of a flexible and self-describing design language. From the previous chapters and a study of ontology it is clear that a design language with a hierarchical structure best facilitates the processing of design knowledge fragments. A successful design is the result of many different cognitive processes at both tacit and explicit levels. These processes can be augmented with many different techniques from the world of manufacturing such as Knowledge Based Design, Systems Analysis, Kansei Engineering, QFD and TRIZ. Many other techniques could be discussed such as FMEA (Failure Mode and Effects Analysis), but it would not contribute significantly to the problem under consideration, the storing of artefact design knowledge over the life cycle of a building and the secondary adaptation of designs.

Once a design is available it can be brought into the ARGOS component (container) to facilitate the positional and shape testing of design fragments. All of this must happen in a neutral environment to guarantee a long life of design information and make it possible for diverse design tools to process relevant parts of the information.

## 6.1 Life cycle Information infrastructure

### 6.1.1 Introduction

Of all the possible candidates investigated Extensible Mark-up Language (XML) proved to be most useful language to solve the stringent requirements for the problem under consideration.

XML, describes a class of data objects called XML documents and partially describes the behaviour of computer programs which process them. XML is an application profile or restricted form of the Standard Generalised Mark-up Language (SGML). XML documents are made of storage units called entities, which contain either parsed or unparsed data. Parsed data is made up of characters, some of which form character data and some, which form the mark-up structure. Mark-up encodes a description of the document’s storage layout and logical structure. XML also provides a mechanism to impose constraints on the storage layout and logical structure. A software module called an *XML processor* is used to read XML documents and provide access to their content and structure.

XML was developed by an XML Working Group formed under the auspices of the World Wide Web Consortium (W3C) in 1996. It was chaired by Bosak of Sun Microsystems with the active participation of an XML Special Interest Group also organised by the W3C.

The primary design goals for XML are:

- XML shall be straightforwardly usable over the Internet.
- XML shall support a wide variety of applications.
- XML shall be compatible with SGML.
- It shall be easy to write programs which process XML documents.

- The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
- XML documents should be in human-legible form and reasonably clear.
- The XML design should be prepared quickly.
- The design of XML shall be formal and concise.
- XML documents shall be easy to create.
- Terseness in XML mark-up is of minimal importance.

### 6.1.2 XML as a design language

Consider Code Fragment 1 below. This is a trivial example of how a materials library could be structured by means of XML. This offers the immediate advantage that the information can be used in other applications and downloaded from the Internet. This structure could be used to implement the Materials Library as detailed in Figure 52 [E1]. In this case the material library starts with the `<MATERIAL_LIBRARY>` label and ends with the `</MATERIAL_LIBRARY>` label. Each separate material starts with the label `<MATERIAL keyword1=”METAL”>` and ends with a `</MATERIAL>`. Hierarchically nested under this is the `<DESCRIPTION>` label that contains a short description of the material.
`<DESCRIPTION>Aluminium (Al) 99.0% pure</DESCRIPTION>`
This is followed by the list of applicable attributes. Note that the attributes are grouped within the attribute label for example:
`<A unit="kg/m3" value="2650.000">Density</A>`
There are no hard and fast rules when to use child elements and when to use attributes. Generally the application developer uses whichever suits his application. A rule of thumb is that data themselves should be stored in elements. Information about the data (meta-data) should be stored in attributes (Harold 1999:101).

Code Fragment 1 could be generated by means of many different methods such as:

- Output from a relational database.
- Dynamic upon demand generation by a web based search engine or query builder.

Domain specific application software such as a design scenario builder could use the basic information contained in the database or could present it in neatly formatted document for reference purposes.

```
<?xml version="1.0" standalone="yes"?>
<?xml-stylesheet type="text/xsl" href="Material_fragment.xsl"?>
        <MATERIAL_LIBRARY>
  <MATERIAL keyword1="METAL">
                <NAME>ALUMINIUM</NAME>
                <DESCRIPTION>Aluminium (Al) 99.0% pure</DESCRIPTION>
                <ATTRIBUTES>
                        <A unit="kg/m3" value="2650.000">Density</A>
                        <A unit="Deg C" value="660.000">Melting point</A>
                        <A unit="N/mm2" minvalue="68300.000">Modulus of elasticity (minimum)</A>
                        <A unit="N/mm2" value="70350.000">Modulus of elasticity (average)</A>
                        <A unit="N/mm2" maxvalue="72400.000">Modulus of elasticity (maximum)</A>
                        <A unit="W/m deg C" value="214.000">Thermal conductivity (k)</A>
                </ATTRIBUTES>
        </MATERIAL>
        <MATERIAL keyword1="METAL">
                <NAME>ALUMINIUM BRONZE</NAME>
                <DESCRIPTION>Aluminium-Bronze Cu 5-10%: Al</DESCRIPTION>
                <ATTRIBUTES>
                        <A unit="kg/m3" minvalue="7570.000">Density (minimum)</A>
                        <A unit="kg/m3" value="2650.000">Density (average)</A>
                        <A unit="kg/m3" maxvalue="8150.000">Density (maximum)</A>
                <A unit="Deg C" minvalue="1041.000">Melting point (minimum)</A>
                <A unit="Deg C" value="1052.000">Melting point (average)</A>
                <A unit="Deg C" maxvalue="1063.000">Melting point (maximum)</A>
                <A unit="N/mm2" value="120000.000">Modulus of elasticity</A>
                <A unit="W/m deg C" minvalue="64.000">Thermal conductivity (k) (minimum)</A>
                <A unit="W/m deg C" value="74.500">Thermal conductivity (k) (average)</A>
                <A unit="W/m deg C" maxvalue="85.500">Thermal conductivity (k) (maximum)</A>
            </ATTRIBUTES>
        </MATERIAL>
        <MATERIAL keyword1="METAL">
            <NAME>BRASS</NAME>
            <DESCRIPTION>Brass Cu 60%: Zn 40%</DESCRIPTION>
            <ATTRIBUTES>
                <A unit="kg/m3" value="8380.000">Density</A>
                <A unit="Deg C" value="904.000">Melting point</A>
                <A unit="N/mm2" value="103000.000">Modulus of elasticity</A>
                <A unit="W/m deg C" value="129.000">Thermal conductivity (k)</A>
            </ATTRIBUTES>
        </MATERIAL>
        <MATERIAL keyword1="WOOD" keyword2="CONSTRUCTION">
            <NAME>PINE</NAME>
            <DESCRIPTION>British Columbia pine</DESCRIPTION>
            <ATTRIBUTES>
                <A unit="kg/m3" minvalue="480.000">Density (minimum)</A>
                <A unit="kg/m3" value="600.000">Density (average)</A>
                <A unit="kg/m3" maxvalue="720.000">Density (maximum)</A>
                <A unit="years" minvalue="10.0">Durability (minimum)</A>
                <A unit="years" value="12.5">Durability (average)</A>
                <A unit="years" maxvalue="15.0">Durability (maximum)</A>
            </ATTRIBUTES>
        </MATERIAL>
</MATERIAL_LIBRARY>
```

Code Fragment 1: Suggested XML structure for the storage of material definitions (Author)

Code Fragment 1 could be formatted, for reporting purposes, at the most basic level by means of Cascading Style Sheets (CSS). CSS styles only apply to XML element content not to attributes in the elements. If CSS were applied to Code Fragment 1 in an Internet Explorer then the attributes would be invisible rendering most of Code Fragment 1 data invisible. However there is an alternative style sheet language that allows the user to access and display attribute data as well. This language is Extensible Style language (XSL). XSL is divided into two main sections:

- Transformations
- Formatting

Consider Code Fragment 2 for an example of a typical XSL that could be used to convert Code Fragment 1 into a neatly formatted output for a web page.

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
    <xsl:template match="/">
        <html>
            <xsl:apply-templates/>
        </html>
    </xsl:template>

    <xsl:template match="/MATERIAL_LIBRARY">
        <html>
            <body>
                <h1>Example Material Library</h1>
                <xsl:apply-templates/>
            </body>
        </html>
    </xsl:template>
```

```
		<xsl:template match="MATERIAL">
			<p>
				<h3><u><xsl:value-of select="NAME"/></u></h3>
				<xsl:apply-templates/>
				<br><hr></hr></br>
			</p>
		</xsl:template>

		<xsl:template match="ATTRIBUTES">
				<xsl:apply-templates/>
		</xsl:template>

		<xsl:template match="A">
			<br><i><xsl:value-of select="."/> = </i>
			<b><xsl:value-of select="@minvalue"/></b>
			<b><xsl:value-of select="@value"/></b>
			<b><xsl:value-of select="@maxvalue"/></b>
			<xsl:value-of select="@unit"/></br>
		</xsl:template>
</xsl:stylesheet>
```

Code Fragment 2: Typical style sheet to format XML data for web page display (Author)

If the XSL in Code Fragment 2 is applied to Code Fragment 1 the output looks like Code Fragment 3. At this stage the XML data in Code Fragment 1 can be used for two entirely different purposes:

- The transfer of structured material attributes for design purposes
- The display of the material characteristics in a web page

## Example Material Library

**ALUMINIUM**

*Density* = **2650.000** kg/m3

*Melting point* = **660.000** Deg C

*Modulus of elasticity (minimum)* = **68300.000** N/mm2

*Modulus of elasticity (average)* = **70350.000** N/mm2

*Modulus of elasticity (maximum)* = **72400.000** N/mm2

*Thermal conductivity (k)* = **214.000** W/m deg C

**ALUMINIUM BRONZE**

*Density (minimum)* = **7570.000** kg/m3

*Density (average)* = **2650.000** kg/m3

*Density (maximum)* = **8150.000** kg/m3

*Melting point (minimum)* = **1041.000** Deg C

*Melting point (average)* = **1052.000** Deg C

*Melting point (maximum)* = **1063.000** Deg C

*Modulus of elasticity* = **120000.000** N/mm2

*Thermal conductivity (k) (minimum)* = **64.000** W/m deg C

*Thermal conductivity (k) (average)* = **74.500** W/m deg C

*Thermal conductivity (k) (maximum)* = **85.500** W/m deg C

**BRASS**

*Density* = **8380.000** kg/m3

*Melting point* = **904.000** Deg C

*Modulus of elasticity* = **103000.000** N/mm2

*Thermal conductivity (k)* = **129.000** W/m deg C

Code Fragment 3: Output generated by Code Fragment 2 applied to Code Fragment 1 (Author)

Consider the formatted output in code fragment 3 that is achieved by means of the XSL code in Code Fragment 2. In this case the output generated from the XML in Code Fragment 1 is HTML format that makes it suitable for direct display in web pages.

Code Fragment 4 below is an example of how the information of a CAD system such as MicroGDS 6.0 is expressed in XML. This system was the first to offer the ability to translate CAD drawings into XML. The XML output is translated into Vector Mark-up Language (VML) by means of a style file. VML is an XML application that combines vector

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MicroGDS PUBLIC "+//IDN informatix.co.uk//MicroGDS600 DTD//EN" "file://c:\usr\PhD\Xml\MicroGDS PhD XML\MicroGDS600.dtd">
<MicroGDS>
<StylePath/>
<Aliases/>
<Styles>
<CVCharstyle Name="18" Height="1.8" FontName="DEFAULT"/>
<CVCharstyle Name="25" Height="2.5" Width="2.5" FontName="DEFAULT"/>
<CVCharstyle Name="35" Height="3.5" Width="3.5" FontName="DEFAULT"/>
<TTCharstyle Name="AR06" Height="6E-1" Width="2.8E-1" FontName="Arial" Weight="Normal" Underline="false" StrikeOut="false" Italic="false" Pitch="Variable"
Family="Swiss"/>
<TTCharstyle Name="AR10" Height="3.5" Width="1.48" FontName="Arial" Weight="Normal" Underline="false" StrikeOut="false" Italic="false" Pitch="Variable"
Family="Swiss"/>
<TTCharstyle Name="AR12" Height="2.5" Width="1" FontName="Arial" Weight="Normal" Underline="false" StrikeOut="false" Italic="false" Pitch="Variable" Family="Swiss"/>
...
<Linestyle Name=".00" Font="DEFAULT" Border="true" Opaque="false" SymbolHeight="2.5" Pen="0" Gap="2"/>
<Linestyle Name=".18" Font="DEFAULT" Border="true" Opaque="false" LeftOffset="4E-2" RightOffset="-4E-2" SymbolHeight="2.5" Pen="0" Gap="2"/>
<Linestyle Name=".25" Font="DEFAULT" Border="true" Opaque="false" LeftOffset="9E-2" RightOffset="-9E-2" SymbolHeight="2.5" Pen="0" Gap="2"/>
...
<Linestyle Name="CENT1" Font="SYMBOL" Border="true" Opaque="false" LeftOffset="5E-2" RightOffset="-5E-2" SymbolHeight="5" Phasing="Line">
<FixedLine Length="7"/>
<EndOfStart/>
<VariableGap Length="1"/>
<VariableLine Length="2"/>
<VariableGap Length="1"/>
<VariableLine Length="2"/>
<VariableGap Length="1"/>
<VariableLine Length="8"/>
<StartOfEnd/>
</Linestyle>
...
<TextMnemonic Name="RGUI" Prompt="Attach  Microsoft global unique identifier" MinLineLength="1"/>
</Styles>
...
<Layer>
<Layer Name="STRUC" Label="" LinkNumber="34" HighestObjectLinkNumber="290">
<Extent LX="0" LY="-2.0927500247955322E2" LZ="0" HX="1.7490625047683716E2" HY="2.0543749952316284E2" HZ="0"/>
...
```

```
<OCD Name="STRUC:STAIR">
<Object LinkNumber="281" HighestPrimitiveLinkNumber="41" Lightstyle="NONE" ContainsItems="false">
<Extent LX="1.0527500009536743E2" LY="-1.2537845563888855E2" LZ="0" HX="1.4290635883501973E2" HY="-
9.4650413990020752E1" HZ="0"/>
<Axes Y="-3" RZ="-7.8539814758300786E-1" S="5E1"/>
<TextPrimitive LinkNumber="41" Mirrored="false" Box="false" Dim="false" Data="false" YFactor="1.1281112432479858"
Charstyle="TNR25" Justification="BC">
<Extent LX="1.2659249997138977E2" LY="-1.2327500009536743E2" LZ="0" HX="1.3083250021934509E2" HY="-
1.1947499952266084E2" HZ="0"/>
<Axes X="1.2871250009536743E2" Y="-1.2327500009536743E2" S="8.8643753592343033E-1"/>
<DefinitionText>UP
</DefinitionText>
</TextPrimitive>
<LinePrimitive LinkNumber="40" Mirrored="false" Linestyle=".00" StartMark="true" EndMark="true">
<Extent LX="1.1563856649398804E2" LY="-1.0827500152587891E2" LZ="0" HX="1.1681673431396484E2" HY="-
1.0694840526580811E2" HZ="0"/>
<Polyline>
<Point X="1.1681673431396484E2" Y="-1.0764382553100586E2"/>
<Point X="1.1576224994659424E2" Y="-1.0827500152587891E2"/>
<Point X="1.1563856649398804E2" Y="-1.0694840526580811E2"/>
</Polyline>
…
</Object>
</OCD>
</Layer>
…
</MicroGDS>
```

Code Fragment 4: Structure of a MicroGDS 6.0 CAD file described with XML (Author)

information with CSS markup to describe vector graphics that can be embedded in Web pages in stead the bitmapped GIF and JPEG images loaded by HTML's IMG element. VML is supported by the various components of Microsoft Office 2000 as well as by Internet Explorer 5.0.

The W3C has received four different proposals for vector graphics in XML from a wide variety of vendors. It's formed the Scalable Vector Graphics (SVG) working group composed of representatives from all these vendors to develop a single specification for an XML representation of Scalable Vector Graphics. When SVG is complete it should provide everything VML currently provides plus a lot more including animation, interactive elements, filters, clipping, masking and pattern fills. A full SVG specification and the software that implements the specification are some time away.

The World Wide Web Consortium released the first working draft of SVG in February 1999 and revised the draft in April 1999. A well advanced working draft appeared 29 June 2000. Microsoft has stated publicly that they intend to ignore any Web graphics efforts except VML.

Code Fragment 4 contains a portion of an XML file that encodes the graphics of a CAD drawing. The XML data starts with the `<MicroGDS>` label and ends with `</MicroGDS>`. In this case the `<StylePath/>` and `<Aliases/>` labels are empty. The next section between the `<Styles>` and `</Styles>` labels defines the various character and linestyles as well as the mnemonics for the attribute data that could be attached to drawing objects (coloured in blue). The particular CAD system under consideration supports both vector type and true type character styles. The former is indicated in a label such as

```
<CV Charstyle Name=”18” Height=”1.8” FontName=”Default”/>
```

and the latter by

```
<TT Charstyle Name=”AR06” Height=”6E-1” Width=”2.8E-1”
FontName=”Arial” Weight=”Normal” Underline=”false”
Strikeout=”false” Italic=”false” Pitch=”Variable”
Family=”Swiss”/>
```

Linestyles could be simple or complex. A typical simple linestyle of .18 mm thickness is described by:

```
<Linestyle Name=”.18” Font=”DEFAULT” Border=”true”
Opaque=”false” Leftoffset=”4E-2” RightOffset=”-4E-2”
SymbolHeight=”2.5” Pen=”0” Gap=”2”/>
```

A more complex linestyle that contains patterns is described by:

```
<Linestyle Name=”CENT1” Font=”SYMBOL” Border=”true”
Opaque=”false” LeftOffset=”5E-2” RightOffset=”-5E-2”
SymbolHeight=”5” Phasing=”Line”>
<FixedLine Length=”7”/>
<EndOfStart/>
<VariableGap Length=”1”/>
<VariableLine Length=”2”/>
<VariableGap Length=”1”/>
<VariableLine Length=”2”/>
<VariableGap Length=”1”/>
<VariableLine Length=”8”/>
<StartOfEnd/>
</Linestyle>
```

The `<TextMnemonic>` label contains the definition of attribute data templates that could be used in this particular case to attach non-graphical information to the graphical objects. In this example a mnemonic called RGUI has been defined. The R in RGUI indicates that the data will apply to a specific instance (reference) of a graphical object. GUI is a mnemonic for Global Unique Identifier. This method has been used in the precedent system AEDES to connect alphanumeric data and graphical data. This particular aspect will have to be developed much further to accommodate the various levels of specificity required for CBR as well as to facilitate constraint propagation, tacit and explicit requirements of design.

The actual graphical data are contained between the `<Layer>` and `</Layer>` labels. In this system graphical data must occur on a layer although it is not a layer-based system. In this case the layer under consideration is `"STRUC"` that indicate that graphical and textual entities related to the structure of the building should be on this layer. The first graphical object is indicated by the `<OCD Name="STRUC:STAIR">` label. This label is closed by the matching `</OCD>` label lower down. Within the bounds of the `<Object>` and `</Object>` labels the graphical text and lines are defined. The part related to text is indicated in blue and the part related to the graphical entities such as polylines in red. The text part is bounded by the `<TextPrimitive>` and `</TextPrimitive>` labels. The polylines are bounded by the `<Polyline>` and `</Polyline>` labels.

The hierarchical nature of the graphical example object conceptually follows the hierarchical structure of:

```
<Layer>
      <OCD>
            <Object>
                  <TextPrimitive>
                  </TextPrimitive>
                  <Polyline>
                  </Polyline>
            </Object>
      </OCD>
</Layer>
```

This forms a useful basis for a design language on which the more extensive requirements of a CBR system that supports design scenario generation and suspension of partially completed designs can be built.

The integrity of Code Fragment 4 is supported by an extensive Document Type Definition (DTD). A DTD provides a list of the elements, attributes, notations and entities contained in a document as well as their relationships to one another. DTDs specify a set of rules for the structure of a document. The DTD accomplishes this with a list of mark-up declarations for particular elements, entities, attributes and notations.

Consider Code Fragment 5 below for a shortened example of a DTD that ensures the integrity of the XML in Code Fragment 4. Only the entities used in Code Fragment 4 are included. The DTD is not necessary if the output is generated by an application. If XML fragments are obtained from other external sources then the DTD ensures conformance to the design language.

```
<!-- Styles -->

<!ELEMENT Styles (
    CVCharstyle|TTCharstyle|
    Linestyle|Material|Lightstyle|
    TextMnemonic|WordMnemonic|DoubleMnemonic|SingleMnemonic|IntegerMnemonic
)*>

<!-- Character Styles
A character style is either defined as a CAD Vector font or a (Windows)
True-Type font.
-->

<!-- Attributes common to character styles -->

<!ENTITY % CharstyleAttributes '
    Name        CDATA           #REQUIRED
    Height      CDATA           #REQUIRED
    Width       CDATA           #IMPLIED
    Pen         CDATA           "1"
'>

<!-- CAD Vector Fonts. The FontName attribute is a font name. -->

<!ELEMENT CVCharstyle EMPTY>
<!ATTLIST CVCharstyle
    %CharstyleAttributes;
    FontName    CDATA           #REQUIRED
>

<!-- True-Type (Windows) font -->

<!ELEMENT TTCharstyle EMPTY>
<!ATTLIST TTCharstyle
    %CharstyleAttributes;
    FontName    CDATA           #REQUIRED
    Weight      (DontCare|Thin|ExtraLight|Light|Normal|Medium|SemiBold|Bold|
                ExtraBold|Heavy)
                                "DontCare"
    Underline   (true|false)    "false"
    StrikeOut   (true|false)    "false"
    Italic      (true|false)    "false"
    Pitch       (Default|Fixed|Variable)
                                "Default"
    Family      (Decorative|DontCare|Modern|Roman|Script|Swiss)
                                "DontCare"
    CharSet     (ANSI|Baltic|ChineseBig5|Default|EastEurope|GB2312|Greek|
                Hangul|Mac|OEM|Russian|ShiftJIS|Symbol|Turkish|Hebrew|Arabic|
                Thai)
                                "ANSI"
>

<!-- Line styles -->

<!ELEMENT Linestyle (
   (FixedLine|FixedGap|Symbol)*,
   EndOfStart,
   (FixedLine|FixedGap|VariableLine|VariableGap|Symbol)*,
   StartOfEnd,
   (FixedLine|FixedGap|Symbol)*
)?>
<!ATTLIST Linestyle
   Name         CDATA           #REQUIRED
   Font         CDATA           #IMPLIED
   VertexStart  CDATA           #IMPLIED
   VertexInternal CDATA         #IMPLIED
   VertexEnd    CDATA           #IMPLIED
   VertexMidPoint CDATA         #IMPLIED
   SegLineStart CDATA           #IMPLIED
   SegLineEnd   CDATA           #IMPLIED
   SegSegStart  CDATA           #IMPLIED
   SegSegEnd    CDATA           #IMPLIED
   FillSymbol   CDATA           #IMPLIED
   Border       (true|false)    "true"
   Opaque       (true|false)    "false"
   LeftOffset   CDATA           "0"
   RightOffset  CDATA           "0"
   SymbolHeight CDATA           "2.5"
   Pen          CDATA           "1"
   Phasing      (None|Angle|Line|Grid)
                                "None"
   Fill (None|HatchHorizontal|HatchVertical|HatchFDiagonal|HatchBDiagonal|
        HatchCross|HatchDiagCross|Solid0|Solid1|Solid5|Solid10|Solid15|Solid20|
        Solid25|Solid30|Solid35|Solid40|Solid45|Solid50|Solid60|Solid70|
        Solid80|Solid90|Solid100|BrushBDiagonal|BrushCross|BrushDiagCross|
        BrushFDiagonal|BrushHorizontal|BrushVertical|FillSymbol)
                                "None"
   Gap          CDATA           #IMPLIED
   Space        CDATA           #IMPLIED
   Shear        CDATA           #IMPLIED
   Slope        CDATA           #IMPLIED
>

<!-- Linestyle pattern elements -->
```

```
<!ELEMENT EndOfStart EMPTY>
<!ELEMENT StartOfEnd EMPTY>

<!ELEMENT FixedLine EMPTY>
<!ATTLIST FixedLine
   Length       CDATA           #REQUIRED
>

<!ELEMENT FixedGap EMPTY>
<!ATTLIST FixedGap
   Length       CDATA           #REQUIRED
>

<!ELEMENT VariableLine EMPTY>
<!ATTLIST VariableLine
   Length       CDATA           #REQUIRED
>

<!ELEMENT VariableGap EMPTY>
<!ATTLIST VariableGap
   Length       CDATA           #REQUIRED
>

<!ELEMENT Symbol EMPTY>
<!ATTLIST Symbol
   Symbol       CDATA           #REQUIRED
>

<!-- Mnemonic definitions -->

<!ENTITY % MnemonicAttributes '
    Name        CDATA           #REQUIRED
    Prompt      CDATA           ""
'>

<!ELEMENT TextMnemonic EMPTY>
<!ATTLIST TextMnemonic
    %MnemonicAttributes;
    MaxLines      CDATA         "1"
    MinLineLength CDATA         "0"
    MaxLineLength CDATA         "132"
>

<!-- Layers -->

<!ELEMENT Layer (Extent?, (Attribute|OCD)*)>
<!ATTLIST Layer
    Name        CDATA           #REQUIRED
    Label       CDATA           ""
    LinkNumber  CDATA           #IMPLIED
    HighestObjectLinkNumber CDATA #IMPLIED
    GUID        CDATA           #IMPLIED
>

<!-- OCD
This is the top-level element for an Object which is a container for the object
name. OCD is short for Object Code (ie name) Definition.
-->

<!ELEMENT OCD (Attribute|Object|ObjectInstance)*>
<!ATTLIST OCD
    Name        CDATA           #REQUIRED
>

<!-- Objects -->

<!ELEMENT Object (Extent?, Axes,
    (Attribute|LinePrimitive|TextPrimitive|RasterPhotoPrimitive|
    WindowPhotoPrimitive|OlePhotoPrimitive|ClumpPrimitive)*
)>
<!ATTLIST Object
    LinkNumber  CDATA           #IMPLIED
    HighestPrimitiveLinkNumber CDATA #IMPLIED
    Lightstyle  CDATA           "NONE"
    ContainsItems (true|false)  "false"
>
```

```
<!-- Line Primitive -->

<!ELEMENT LinePrimitive (%PrimitiveContent;, Polyline)>
<!ATTLIST LinePrimitive
     %PrimitiveAttributes;
    Linestyle   CDATA           "DEFAULT"
    StartMark   (true|false)    "true"
    EndMark     (true|false)    "true"
>

<!-- Text Primitive -->

<!ELEMENT DefinitionText (#PCDATA)>
<!ELEMENT ExpandedText   (#PCDATA)>

<!ELEMENT TextPrimitive (
    %PrimitiveContent;, Axes,
    DefinitionText, ExpandedText?
)>

<!ATTLIST TextPrimitive
    %PrimitiveAttributes;
    Charstyle   CDATA           "DEFAULT"
    Linestyle   CDATA           #IMPLIED
    Justification (TL|TC|TR|CL|CC|CR|BL|BC|BR)
                                "BL"
    Box         (true|false)    #IMPLIED
    Dim         (true|false)    #IMPLIED
    Data        (true|false)    #IMPLIED
    YFactor     CDATA           "1"
>

<!-- Points simply consist of x,y,z coordinates -->

<!ELEMENT Point EMPTY>
<!ATTLIST Point
    X           CDATA           "0"
    Y           CDATA           "0"
    Z           CDATA           "0"
>

<!-- As far as the DTD is concerned, a vector is equivalent to a point -->

<!ELEMENT Vector EMPTY>
<!ATTLIST Vector
    X           CDATA           "0"
    Y           CDATA           "0"
    Z           CDATA           "0"
>

<!-- BulgeAxis - this is a bulge factor
The B attribute represents the bulge factor, a value between 0 and 1. -->

<!ELEMENT BulgeAxis EMPTY>
<!ATTLIST BulgeAxis
    B           CDATA           "0"
    X           CDATA           "0"
    Y           CDATA           "0"
    Z           CDATA           "0"
    A           CDATA           #IMPLIED
>

<!-- Polyline
Start point, followed by a sequence of line segments (curved or straight).  If
the first and last points are the same, the polyline is closed.
-->

<!ELEMENT Polyline (Point, (BulgeAxis?, Point)*)>
```

Code Fragment 5: Partial MicroGDS 6.0 XML Document Type Definition (DTD) described with XML (Author)

At this stage it is possible to implement the conceptual design processor illustrated in Figure 46.

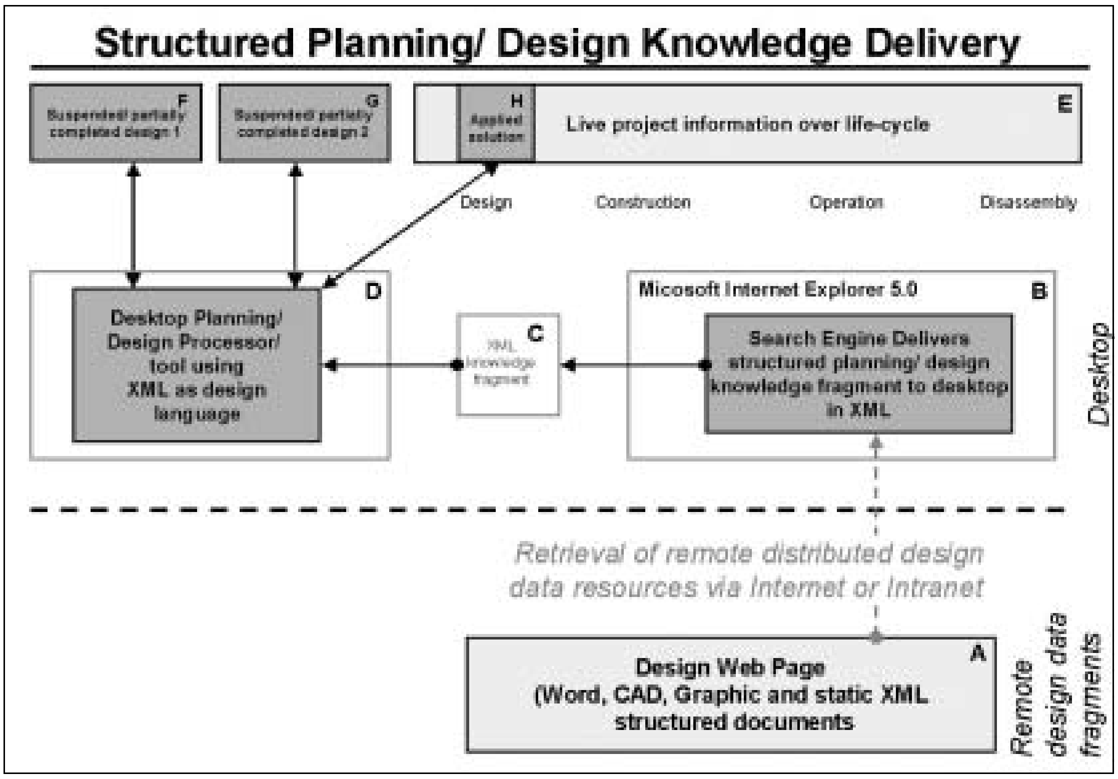

Structured Planning/ Design Knowledge Delivery
F
Suspended partially completed design 1
G
Suspended partially completed design 2
H
Applied solution
Live project information over life-cycle
E
Design
Construction
Operation
Disassembly
D
Desktop Planning/ Design Processor/ tool using XML as design language
XML knowledge fragment
C
Micosoft Internet Explorer 5.0
B
Search Engine Delivers structured planning/ design knowledge fragment to desktop in XML
Desktop
Retrieval of remote distributed design data resources via Internet or Intranet
A
Design Web Page (Word, CAD, Graphic and static XML structured documents
Remote design data fragments

Figure 1: Structured Planning/ Design Knowledge Delivery (Author)

The design knowledge delivery system will conceptually work as detailed in Figure 46. A designer that wants to design a facility or solve a specific operational problem will activate a purpose made search engine [B] in Microsoft Internet Explorer. The search engine [B] will enable the user to set basic constraints and search criteria in order to expedite information retrieval. If the relevant information is found it will be packaged in the form of XML design knowledge fragments. The user can first view the result in Internet Explorer and if he is satisfied ask the system to download it to the desktop. The desktop planning/ design processor [D] will retrieve the downloaded XML knowledge fragment [C]. Due to the fact that design takes place in an open world it is expected that many different planning concepts might exist that need to be explored. These partially completed scenarios are stored in [F] and [G] again in XML format. Once the planner is satisfied the solution can be plugged into a live project environment [H]. It is also possible to publish good designs back into an office web page [A] to make them available to other designers.

[D] could be seen as *working memory* (WM), [F] and [G] as *long term memory* (LTM) (Simina 1999:39-43). The main purpose of WM is:

- Promote synergy among design parts
- WM facilitate external and internal event detection and processing
- WM keeps a limited store of recently accessed artefacts

The purpose of  LTM is:

- Main repository of past design or design fragments
- Retrieval from LTM could be based on any combination constraints or functions

The XML Fragment interchange working draft (W3C 1999) defines a way to send fragments of an XML document to an XML user, in this case the designer using the desktop/ planning processor. It must be emphasised that although Figure 46 is an oversimplified example the following important principles are used:

- The designer remains in full control of the ultimate solution at all times
- Design experience is stored in a structured format (the beginning of CBR)
- Most information required in the planning and design environments are basically hierarchical and occur at various levels of specificity
- XML supports the inclusion of non-XML data and can act as an integrator of diverse data sources
- XML supports distribution of data as well as data hyper linking
- XML supports multi-media data sources
- The example attempts to support design as a pragmatic as well as a cognitive activity
- The solution assumes that planning and design requires a continuum of design methods that use model based, rule based and case-based reasoning. It is ultimately up to the designer to decide what method he prefers
- Current relational databases such as Oracle already support the generation of XML data from a relational query

By means of a style sheet defined in XSL it is possible to display the XML such as Code Fragment 4 in vector format in a web page (Figure 47). For a complete listing of the style sheet please consult Appendix D. The style sheet converts the XML code into Microsoft VML format that makes the display in a web page possible.

The display as illustrated in Figure 47 was done in the smallest possible custom developed web browser for the following reasons:

- To test the feasability of a thin browser developed in Visual Basic by means of the convenient *Inet* ActiveX control.
- To facilitate retrieval of XML code fragments in the ARGOS autonomous design objects a small Internet Explorer is required that has the ability to interpret the XML, stylesheets and DTDs.

The actual code required to implement the minimal browser is included in Appendix F. The browser was tested by means of a small test web page run on a personal computer by means of the Personal Web Server provided with Microsoft Windows 98. Only minimal functionality is provided but enough to facilitate connection to any potential design site in the world or design knowledge fragment on the personal machine or Intranet.

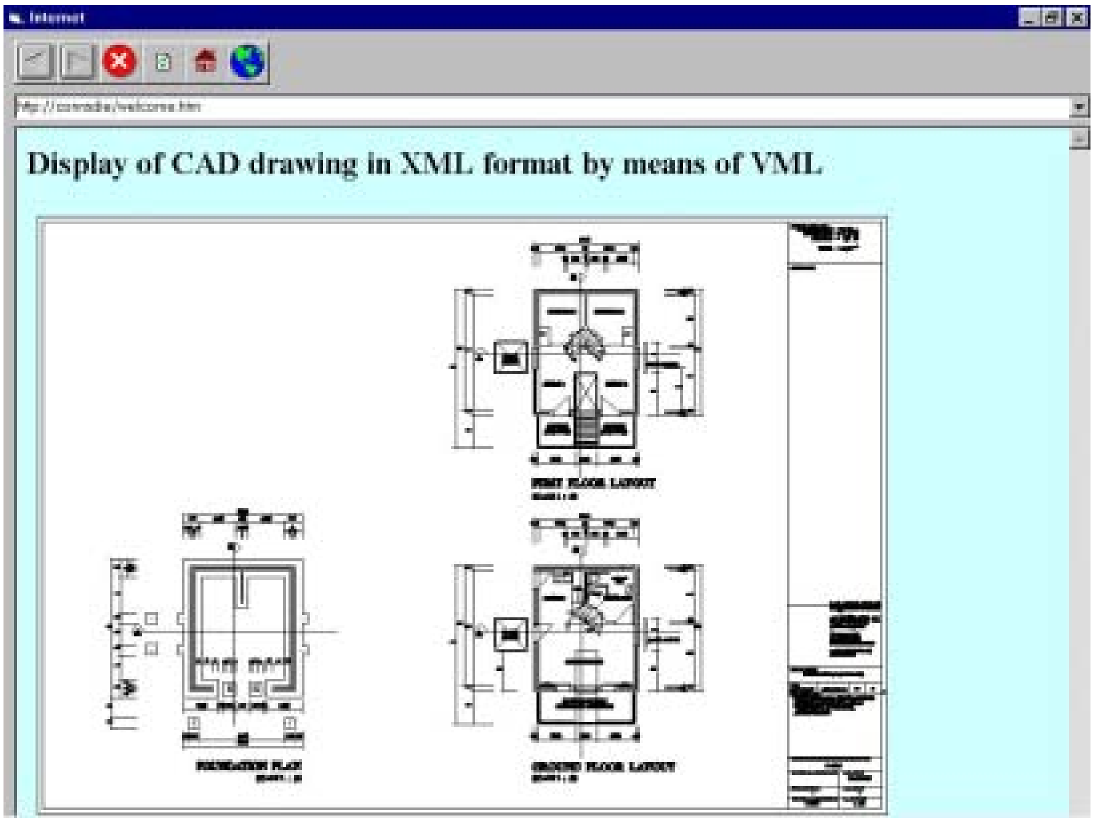

Internet
http://conradie/welcome.htm
Display of CAD drawing in XML format by means of VML
FIRST FLOOR LAYOUT
FOUNDATION PLAN
GROUND FLOOR LAYOUT

Figure 2: Display of CAD drawing in XML format by means of VML (Author)

Careful analysis of the display reveals numerous small errors such as inaccurate text display, and problems with the interpretation of the bulge factor to display circles or arcs. Bit map images, although saved in the XML file are not displayed at all in the web page. At this stage the display capabilities of an AutoCAD Whip file in the web environment are superior to what is offered by the static VML display. However the XML provides a structured and accessible data format that can be processed further whereas the Whip[1] format is closed and proprietary.

## 6.2 Packaging and retrieval of design knowledge

### 6.2.1 Introduction

The author proposes a totally new approach to architectural design knowledge packaging that would require the lowest possible level of platform technology, such as a spreadsheet, as the entry level. Many ambitious attempts have been made in the past to define universal Building Product Models. At this stage none of them are entirely satisfactory due to complexity of the artefact creation world. All indications are that conscensus will be reached soon (Eastman 1999)

The portable nature of Microsoft ActiveX controls makes it possible to support a wide range of platforms without being tied into particular CAD systems, databases or design software. It also ensures a cost effective design environment. By means of ActiveX controls that are embedded into web pages it is possible for service providers to offer a subscription service of design tools such as lightweight cases (architectural design kits) to designers. The designer could then use design software in his Internet explorer without even installing or buying

[1] Whip is a proprietary format that facilitates the display of CAD drawings in a web page

expensive software. The user could then purchase time from the software service provider only when required. It is proposed that the approach that has been followed in the development of the precedent systems AEDES and PREMIS up to date be completely changed around. The approach in the past was an application centric approach with particular emphasis on specific database technology and CAD systems. It is proposed to use a document-based approach (Figure 52). Designers and Architects are used to the concept of documents. This will ensure that anybody that has Microsoft OLE, COM and DCOM compliant software can significantly benefit from the approach. Microsoft developed these technologies specifically to support the intelligent use of documents in a collaborative environment.

The architectural design starter kits, developed by the Division of Building Technology, over a long period of time provided a useful starting point for AEDES and the present research. These starter kits have already contributed significantly towards an accelerated and more efficient design process in the domain of health facilities. These starter kits are available in CAD format and contains "empirical ideal" total facility layouts. The author recently wrote a prototype web page to test the technical feasibility of the distribution of these design kits via the Internet.

The main shortcomings of the present CSIR starter kits (cases) are:

- They contain no traceability of the design process.
- Although staffing required, fixed and loose equipment are available in supporting design documents, it is not in a structured way that could be used in Case-Based Reasoning.
- No distinction is made between neutral and localised information. Heidegger described this as the *"Dasein"* of tools (Biemel 1976:38).
- No object naming conventions have been used that can facilitate connection with other data sources.
- No intelligence is available to predict operational performance.
- No integration with the total life cycle information infrastructure.
- Starter kits should contain knowledge from both the tacit and explicit levels of knowledge management to give future users an idea what the design rationale was.

The use of structured methods such as QFD, Kansei and System Engineering is explicitly time-consuming (Cohen 1995:31). In order to achieve the best possible future use of the design knowledge it is important that knowledge can be reused. The object technologies presently available are already mature enough to support this need well.

The AEDES prototype software solved some of the abovementioned knowledge packaging challenges. However a few fundamental matters are still unresolved such as support for the Internet, complete object encapsulation and a low-level entry platform. In the prototype CAD drawings were embedded into an OLE field into an Oracle form field in an attempt to encapsulate the various types of knowledge required. The main disadvantages of this approach were:

- The object data is not persistent.
- Low-level users would require a database such as Oracle, Microsoft Access or SQL Server as a minimum to use the starter kit.
- It would be difficult to distribute the design globally.
- The response by means of Visual Basic during interrogation of the embedded CAD objects is presently very slow. This improved significantly in Oracle 8.0i (The latest Oracle RDBMS release).

- It is difficult and inconvenient to interface the alphanumeric and graphic contents of the starter kit with other applications.
- It would be difficult for third party companies to build starter kits independently. This is a prerequisite if the starter kits are to gain widespread commercial acceptance in future.
- It is particularly difficult to profile or deliver design data to suit the specific needs of the designer, planner or reasoner.

### 6.2.2 Constraints

Constraints form an important part of planning and design in general and should be supported by ARGOS. The following different main categories of constraints can be identified that could be supported by ARGOS:

- *Formulation*. This is the process of adding or creating new constraints based on decisions. Constraints could originate from the designer, propagation of second order constraints and by inheritance.
- *Propagation*. This is the process of inferring values and constraints from other values and constraints. This is achieved by means of functions associated with the constraint type.
- *Satisfaction*. This is the process of finding values that satisfy a constraint set. Different constraints can have different functions associated with the particular constraint.

The implementation of constraints in an open world is subject to several requirements:

- *Sensitivity to incomplete knowledge*. It is possible that constraints need to be evaluated with some arguments missing. Hinrichs (1991:98) suggests that two evaluation functions are used in this situation, one that is optimistic about the missing information and one that is pessimistic.
- *Ability to relax preferences*. Since design problems may have satisficing solutions, the design processor needs to be able to relax constraints. To facilitate this the importance of a specific constraint needs to be known.
- *Flexibility of propagation*. The constraint poster should be able to propagate constraints between different sets of variables in a problem.
- *Protection of problem-independent constants*. The flexibility of propagation necessitates the restriction of what counts as a variable in a problem.

The constraints determine the class of problems that can be represented. Figure 48 illustrates the taxonomy of constraint types that could be used in a design processor. The constraints fall into five main categories:

- *Logical Connectives* permit recursive combinations of constraints.
- *Nominal Constraints* relate identities of values.
- *Ordinal Constraints* capture relationships between continuous valued quantities.
- *Structural Constraints* constrain the existence of variables rather than their values.
- *Functional Constraints* degenerate constraints (rules) that propagate only in one direction.
- *Second-Order Constraints* are constraints on other constraints.

In the descriptions below, the term *variable* refers to a slot in some frame and *argument* refers to an actual argument to the constraint, which could be either a variable or a constant (Figure 48).

*Same*. Two arguments are constrained to be identical. This is typically used to connect two variables together. It could also be used to restrict a variable to a constant.

*Instance*. The first argument must be a frame subtype of the second. In this case instances and subtypes are treated equivalently.

*Compatible*. The arguments must be frames, in which neither is represented as being *incompatible* with the other.

*Inverse*. The first argument must be the logical or functional inverse of the second.

*Member*. The first argument is a member of the set designated by the second argument.

*Contains*. The second argument is an ingredient or component of the first argument. This is a transitive relationship.

*Does-not-Contain*. The second argument is not an ingredient or component of the first argument. This is a transitive relationship.

*Within*. The first argument is in the numerical range designated by the second argument.

*At-least*. The first argument is greater than or equal to the second.

*At-Most*. The first argument is less than or equal to the second.

*Max*. The first argument is the maximum of all subsequent arguments.

*Min*. The first argument is the minimum of all subsequent arguments.

*Same-Structure*. The variables in the first argument are the same as the variables in the second argument. The arguments to structural constraints of this sort are effectively quoted[1] such that variables themselves are returned, rather than the values of those variables. This permits constraints on structure as well as on content.

*Struc-Member*. The variable in the first argument is a member of the variables in the second argument.

*Same-Constraints*. Every constraint on the internal slots of the value of the variable is present on the corresponding slots of the frame containing the variable.

*Constraints*. The constraints on the first argument are propagated to all the variables designated by the second argument.

To carry out a planning and design activities certain information must be available. In addition, certain conditions, states or evaluations may apply to the data. Eastman (1999: 343) calls this the *Readset* and *Before Constraints*. When an activity is completed data will be added or modified. That design data will possibly have new conditions, constraints or states associated with it. Eastman call this the activities' *Writeset* and *After Constraints*. Together they define an activity $\Phi$ that has the following general structure:

$$\Phi = (\{E\}^R, \{E\}^W, \{C\}^B, \{C\}^A)$$

| | | |
|---|---|---|
| where | $\{E\}^R =$ | the set of entities to be read into the application |
| | $\{E\}^W =$ | the set of entities that are written by the application |

1 This term refers to a convenient LISP construct. LISP was a prominent language in AI ten years ago.

| | |
|---|---|
| $\{C\}^B =$ | the set of constraints that must be satisfied before the application can be executed |
| $\{C\}^A =$ | the set of constraints that are satisfied within the application and can be relied on by later operations |

The before constraints and after constraints specify a logical relationship between activities and the information and the conditions that the activities require. The *Readsets* and *Writesets* define data dependancies. The before constraints and after constraints identify process dependencies.

Constraints, as defined here, can have one of four values (Eastman 1999:343):

| | |
|---|---|
| $< T >== True$ | implies that it has been satisfied |
| $< F >== False$ | implies that it has been evaluated and has failed |
| $< U >== Unknown$ | implies that it has not been evaluated, possibly because it is not available to do so |
| $< X >== Blank$ | implies that changes have been made to the context, so that the state of the constraint is uncertain |

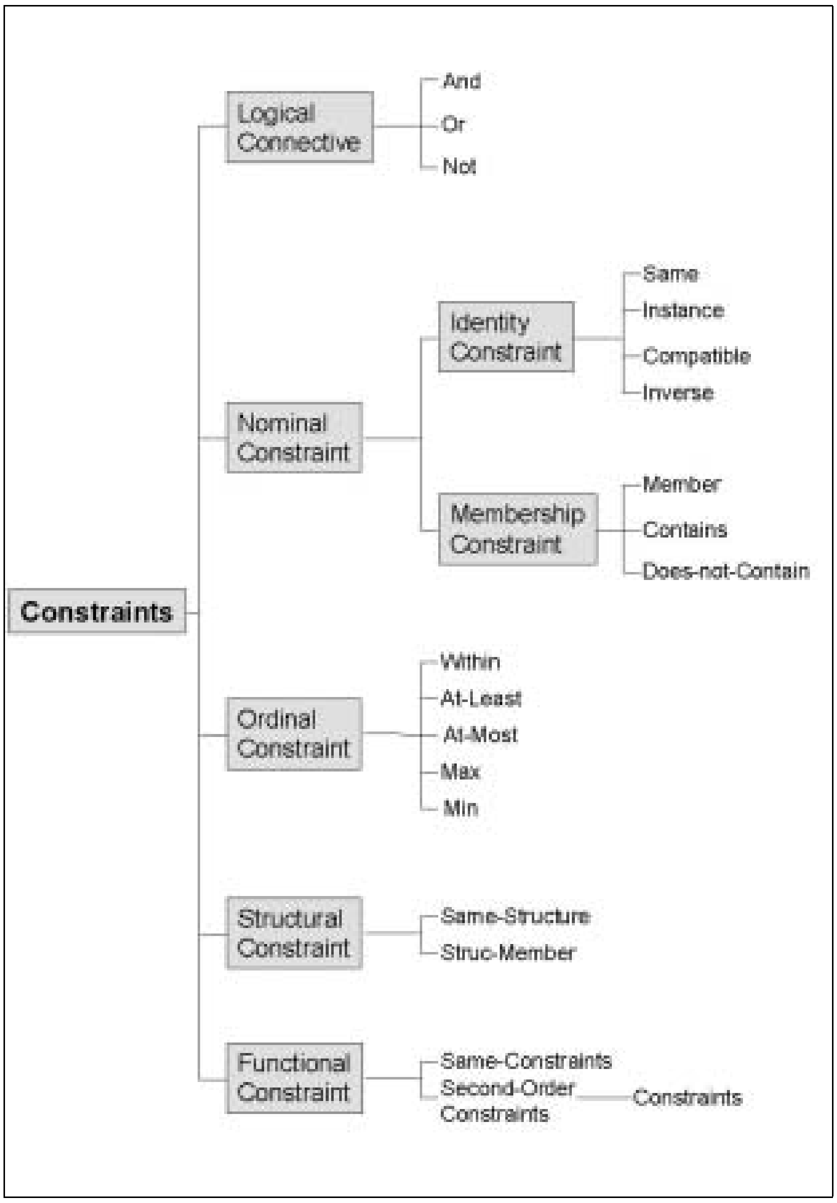


Figure 3: Taxonomy of constraint types (Hinrichs 1991:99)

### 6.2.3 The design of the ARGOS intelligent component

In order to conveniently process design fragments on the desktop without the use of CAD requires intelligent components that can encapsulate the design fragments. The component should also be able to retrieve design fragments from anywhere. To test the idea a prototype control was built. Consider Figure 49 for an example of the control running in Internet Explorer 5.0 The component has the ability to be resized in the x and y axis whilst in two dimensional mode and the x and z axis whilst in three dimensional mode. Appendix F contains the actual code that connects the various parts of the control parametrically together. The intention is that a designer might place a number of the controls in a spreadsheet to test the relationship between architectural design units at any level of specificity. Each component is an autonomous encapsulated world on its own.

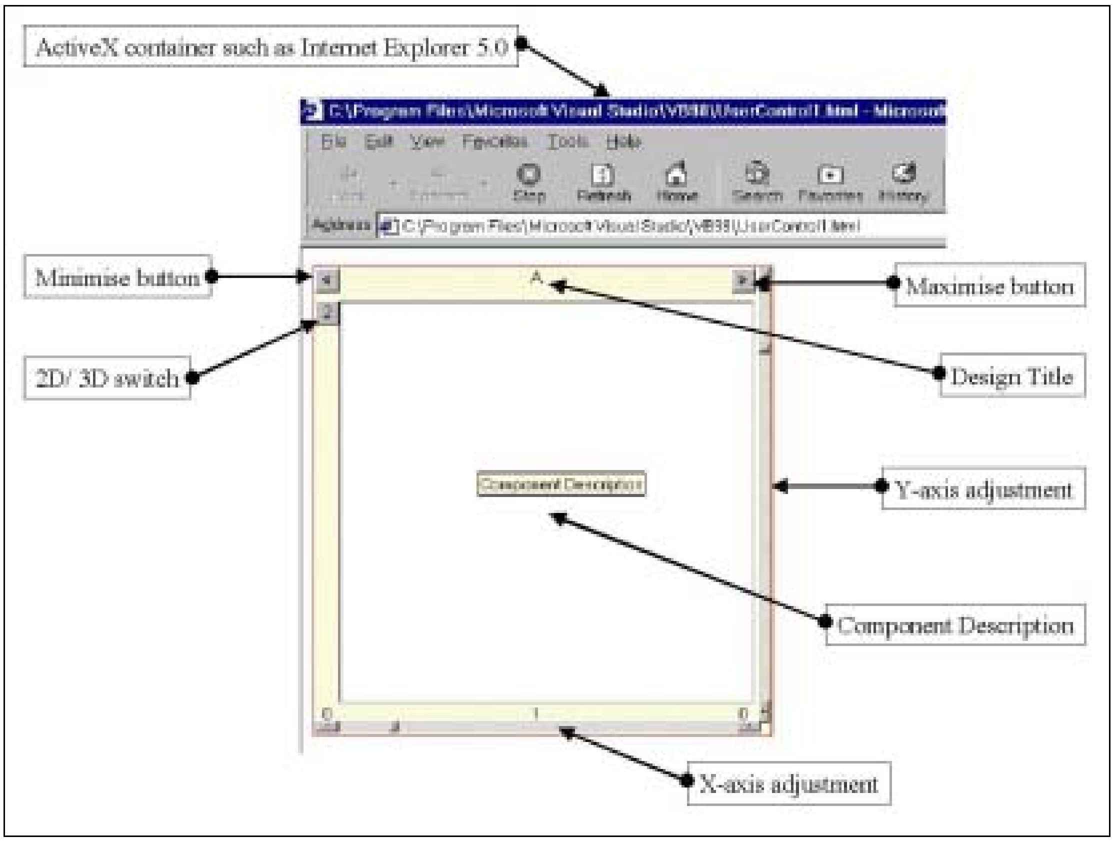

ActiveX container such as Internet Explorer 5.0
C:\Program Files\Microsoft Visual Studio\VB98\UserControl1.html - Microsoft
File Edit View Favorites Tools Help
Stop Refresh Home Search Favorites History
Address C:\Program Files\Microsoft Visual Studio\VB98\UserControl1.html
Minimise button
Maximise button
2D/ 3D switch
Design Title
Component Description
Y-axis adjustment
Component Description
X-axis adjustment

Figure 4: ARGOS object in 2D mode (Author)

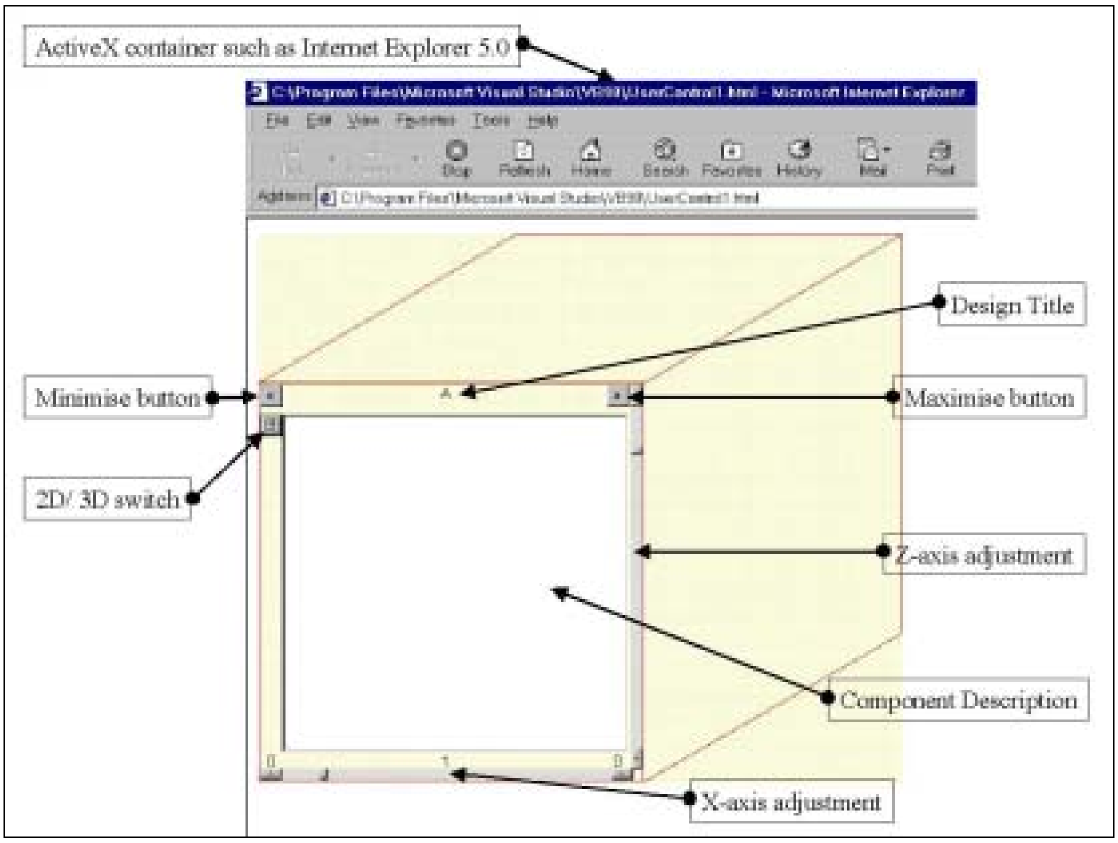

ActiveX container such as Internet Explorer 5.0
C:\Program Files\Microsoft Visual Studio\VB98\UserControl1.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Stop Refresh Home Search Favorites History Mail Print
Address C:\Program Files\Microsoft Visual Studio\VB98\UserControl1.html
Design Title
Minimise button
Maximise button
2D/ 3D switch
Z-axis adjustment
Component Description
X-axis adjustment

Figure 5: ARGOS object in 3D mode (Author)

Synchronisation between the autonomous components can achieved by means of very simple Visual Basic for Application code if it is used in a spreadsheet.

Figure 50 illustrates the ARGOS control in 3D mode. The 3D mode enables a designer to get a feeling for volume in a basic way. The z-axis adjustment facilitates the adjustment of the height. The controls can be made highly sophisticated by adding automatic volume calculation and readout of pertinent design parameters. It is envisaged that many different variants of ARGOS can be built such as:

- Controls that are unlocked, leaving it up the user to place, size and populate them with design information
- Controls that contain cases from the past at various levels of specificity
- Rule-Based controls that model certain well known design characteristics such as energy use
- Model-Based controls that model the constructional performance of a structure such as the forces on a slab

### 6.2.4 Classification and knowledge organisation in a packaged environment

If the packaging of architectural design knowledge in the form of encapsulated Microsoft ActiveX controls is to be successful then it is important that a designer can easily find relevant controls anywhere in the world. It is also important to realise that a control that has not been brought into the specific environment where it will be used should contain knowledge that is neutral. Once it arrives in the specific environment where it will be used it should take on the localised qualities. An example of this is the cost of plant, labour, construction materials, temperature and soil conditions.

The core problem in Information Science (IS) is seen as information seeking and “information retrieval (IR). The design of information systems and knowledge organisation by classification and indexing is a means to that end.

Hjörland (ISKO 1994:91) identifies nine principles on the organisation of knowledge.

1. Naïve-realistic perception of knowledge structures is not possible in more advanced sciences. The deepest principle on the organisation on knowledge rests upon principles developed in and by scientific disciplines.

2. Categorisations and classifications should unite related subjects and separate unrelated ones. In naïve realism, subject relationships are based on similarity. Two things or subjects are seen as related if they are “alike”, that is if they have common properties or descriptive terms ascribed.

3. For practical purposes, knowledge can be organised in different ways and with different levels of ambition.

    - Ad-hoc classification (categorisation) reflects a very low level of ambition in knowledge organisation. Every time you arrange flowers in your private home, you use a kind of “ad-hoc classification” determined by your private taste, the colours of your rooms, what other objects they should match with.
    - Pragmatic classification reflects a middle level of ambition in knowledge organisation. It is a compromise between ad hoc classifications and scientific classifications. Amateur gardeners or horticulturists have other criteria for categorising proteas and azaleas than the biologist would imply.

- Scientific classification reflects a very high level of ambition in knowledge organisation. It is highly abstract and generalised way of organising knowledge. An example of this is the classification of animals and plants according to biological taxonomies.

4. Any given categorisation should reflect the purpose of that categorisation. It is very important to teach the student to find out the lie of the land and apply ad hoc classifications, pragmatic classifications or scientific classifications when appropriate.

5. Concrete scientific categorisations and classifications can always be questioned. The concept of “science” has more than one meaning.

   - Science as a social institution, consisting of people paid to do research. This is the cultural concept of science.
   - Science as a normative, epistemological concept (to argue in a scientific way). What constitutes science in this respect is a matter of continuous development, argument and criticism in methodology and theory of science and in the development of science itself.

6. The concept of “polyrepresentation” is important. In typical information seeking situations, some categorisations are useful to some degree, others to some other degree.

7. To a certain degree different arts and sciences could be understood as different ways of organising the same phenomena.

8. The nature of disciplines varies. The distinction between “hard sciences” and “soft sciences” is well known, but perhaps not fruitful.

9. Many authors indicated the important problem that the quality of knowledge production in many disciplines is in great trouble. It seems if the priorities become more and more short-sighted, that less effort are made to develop long-sighted, well organised and well-cared for bodies of knowledge and literature. This means, that the integrity of scientific knowledge as well as other forms of knowledge is threatened.

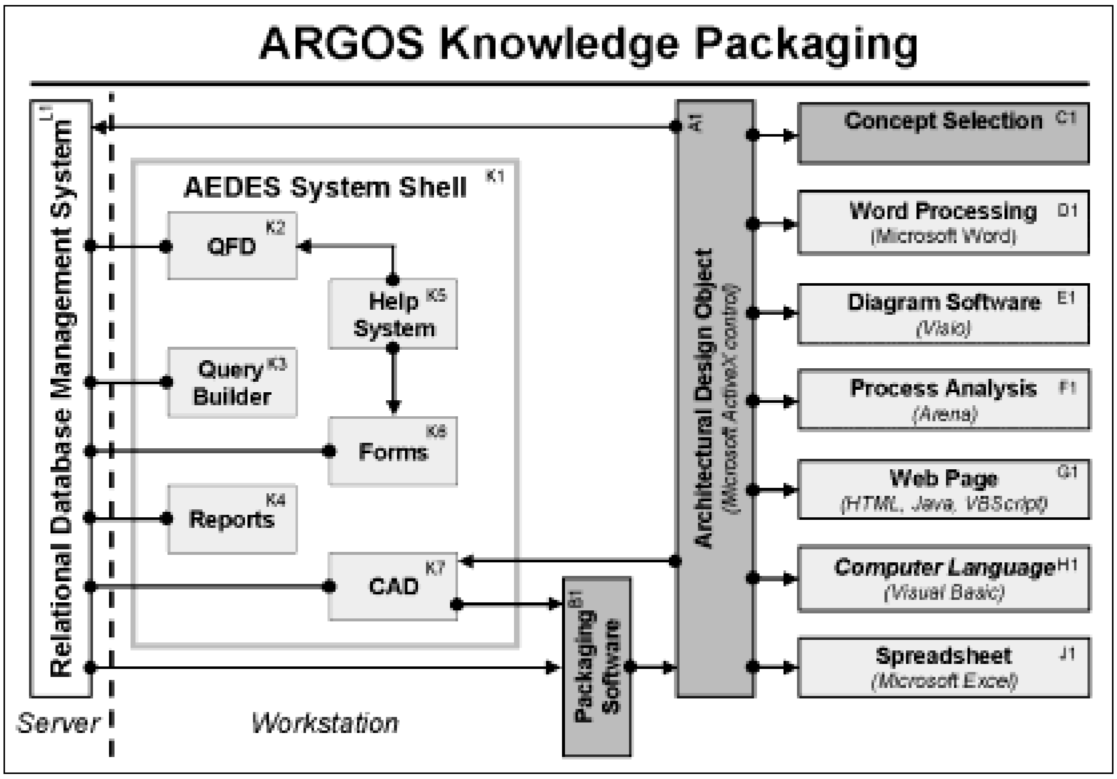

ARGOS Knowledge Packaging
Relational Database Management System L1
AEDES System Shell K1
QFD K2
Query Builder K3
Reports K4
Help System K5
Forms K6
CAD K7
Packaging Software B1
Architectural Design Object (Microsoft ActiveX control) A1
Concept Selection C1
Word Processing (Microsoft Word) D1
Diagram Software (Visio) E1
Process Analysis (Arena) F1
Web Page (HTML, Java, VBScript) G1
Computer Language (Visual Basic) H1
Spreadsheet (Microsoft Excel) J1
Server
Workstation

Figure 6: The relationship between the ARGOS, ActiveX design object and the applications software (Author)

### 6.2.5 The co-existence of ARGOS with other software

The Architectural General Object System (ARGOS) is a Microsoft ActiveX object with the internal design fragment stored in XML. Microsoft Visual Basic provides enough functionality to build the object (Appleman 1999). Although third party users can generate the object independently, it is recommended that a structured front end consisting of an appropriate collection of methodologies as described for the AEDES system could be used. This will ensure that the object is optimal for the given set of requirements. COM software components such as ActiveX controls can be developed with several different programming languages. The most common choice, if Web pages on the subject are any indication, is Microsoft Visual C++. Presently Visual Basic 5 and 6 also support the development of ActiveX controls very well. Using Visual C++, COM software can be written using one of three development libraries, the ActiveX Template Library, Microsoft Foundation Class Library or the BaseCtl framework. ActiveX controls can use a variety of programming languages from Microsoft for component design in addition to Visual C++ like Visual Basic, Visual J++ and even Word or Excel's programming languages.

Currently only highly skilled programmers can build the ActiveX objects. For this reason a special module B1, Packaging Software is proposed (Figure 51). This software tool takes the final design fragment and encapsulates it into a single object. Once the object is created it can be distributed in many different ways and used in a wide variety of environments. The contents of the object can be imported back into the original environment that created it. However the object can be used in many other environments such as spreadsheets, Web pages and process analysis.

### 6.2.5.1 Concept selection

The process of concept selection is important in the product development environment and therefore architectural design. In Architecture it is often necessary to compare alternative architectural design concepts, especially during the early phases of design. To this end the ARGOS kits could be inserted into a spreadsheet. The designer could then conveniently analyse various design aspects in the familiar environment of a spreadsheet without doing any programming. In this case the controls containing the likely concepts would be drawn into a spreadsheet or a simple Visual Basic program. The ratings from the different concepts are derived from the controls and subsequently compared with one another.

### 6.2.5.2 Spreadsheets

Spreadsheets such as Microsoft Excel support the use of ActiveX controls. Many people use spreadsheets and it is a convenient environment for initial project planning tasks such as cost estimating, area and energy analysis. In this environment there is no need to be connected to a database, although the proposed design of the ARGOS object includes links to material and product databases.

In order to use the control in this environment, a user simply has to insert the control into the spreadsheet. To access the list of properties and methods provided in the control in the spreadsheet, the user has to connect the desired property in the control to a cell(s) in the spreadsheet. This can be achieved by the example code fragments below (Code Fragment 3).

In this case the cells are manipulated by the Visual Basic *GotFocus* and *LostFocus* events. In the case of the function *ArgosAB_GotFocus* a range of cells *Range("A1:A10")* on *Worksheet("Sheet1")* is set to the value of the *GrossArea* property retrieved from object instance *ArgosAB*. Note that during the creation of the object certain properties were set to read only. In a similar way the function *ArgosAB_lostFocus* sets the *value* of a range of cells *Range("A1:A10")* to an empty string. At the same time a property *text* of the text box *txtArgos* is set to the text string "RESET TO EMPTY". The control is a totally encapsulated world that contains many properties. These autonomous controls need to be connected together in order to do something useful with it. This can be achieved in any ActiveX compliant container environment.

```
Private Sub ArgosAB_GotFocus()
    Worksheets("Sheet1").Range("A1:A10").Value = ArgosAB.GrossArea
    txtArgos.Text = ArgosAB.GrossArea
End Sub

Private Sub ArgosAB_LostFocus()
    Worksheets("Sheet1").Range("A1:A10").Value = ""
    txtArgos.Text = "RESET TO EMPTY"
End Sub
```

Code Fragment 3: Communication between an ARGOS ActiveX control and Excel Spreadsheet cells (Author)

### 6.2.5.3 Computer languages

A systems integrator or software tool designer can use the ActiveX controls (objects) in exactly the same way. However he can implement the objects in far more advanced environments. A typical scenario would be where a suggested method such as *ArgosAB.UnpackFunction* or *ArgosAB.UnpackCAD* could be invoked. This tells the particular instance of the design component (*ArgosAB*) that the user wants to inspect the particular design functions embodied into the design or want the design object to download the CAD drawing to start with CAD based layout planning.

#### 6.2.5.4 Process analysis

In an environment such as offered by *Arena* users can use the control to extract the desired properties that he wants to analyse. The capabilities of *Arena* can be utilised to optimise flow of people in the specific layout. *Arena* uses Visual Basic for Applications (VBA) as its command language.

### 6.2.6 The design of the ARGOS object

The ARGOS object [A1] can be placed inside any ActiveX container [B1] such as supported by Excel, Word, Visio or World Wide Web pages. Due to the intrinsic information that is built into the object the designer can use the object immediately without connecting to any outside information sources. However to realise the full power of this approach it is recommended that a user connects to the Internet to access convenient outside data sources to provide information such as product data [D1], material characteristics [E1], other existing cases [F1] and Facilities Management cost models. Figure 52 illustrates this concept as well as the relationship of the object with such remote data sources.

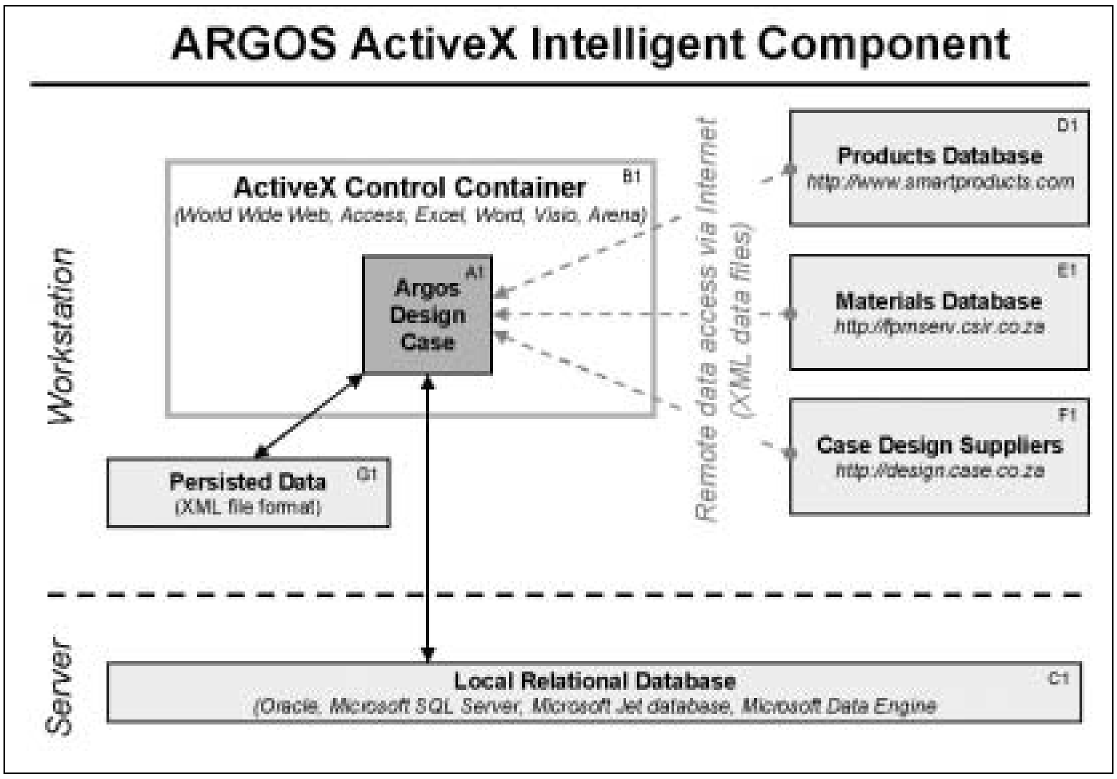

ARGOS ActiveX Intelligent Component
Workstation
Server
ActiveX Control Container B1
(World Wide Web, Access, Excel, Word, Visio, Arena)
Argos Design Case A1
Persisted Data G1
(XML file format)
Remote data access via Internet (XML data files)
Products Database D1
http://www.smartproducts.com
Materials Database E1
http://fpmserv.csir.co.za
Case Design Suppliers F1
http://design.case.co.za
Local Relational Database C1
(Oracle, Microsoft SQL Server, Microsoft Jet database, Microsoft Data Engine

Figure 7: The relationship of the ARGOS object to other intelligent data sources (Author)

Internally the ARGOS design case contains 4 main types of design knowledge that consists of both alphanumeric and graphic information:

- Tacit design information
- Explicit design information

- Graphic information in the form of a design drawing that should preferably be in a neutral data format
- Functional design information such as the design functions and their allocation to physical design elements in a structured format. The W3C, XML format is ideal for this

Some of the information in the first two groups is exposed directly as ActiveX properties. In this case the design object properties is a synonym for surface features[1]. The indices of a case are those combinations of features that distinguish it from other cases, because they are *predictive* of something important in the case. In addition to be being predictive of something important, indices need to be *concrete* enough to be recognisable and *abstract enough* to make a case useful in a variety of future situations. This enables a designer to assess the applicability of the design or to estimate approximate cost. If the design appears to be suitable then the detailed functional design can be inspected.

Again it is important to note that it is an incorrect assumption when people argue about surface features, deep features, structural features, pragmatic features and thematic features in the sense of designing retrieval methods for cases based on one of those. To build a good index it is important to choose from all these levels and make sure that it has the important properties (Kolodner 1996:357). Those descriptors describe where a feature lies in a representation or what its content is. Sengupta *et al.* (1999) gives an indication of the usefulness of the W3C, XML standard for the representation of a case structure and describes methods to translate between relational databases and XML. The author is of the opinion that XML is almost ideal for the structured documentation of the intrinsic artefact design functions. By structured the following is assumed:

- The structure can be analysed by means of computer software.
- It is a complete documentation of the design *performance requirement*, *functions*, *allocations* to construction elements and *specifications* using systems engineering principles.
- Design function groups can be inserted into the existing structure.
- Constraint posting can be supported.
- Quality is an intrinsic part of the function structure.

If adaptation is required then the functional tree can be modified. Modification could be by means of the insertion of function fragments, elements or specifications. If existing design fragments cannot be found then the designer has to design the specific parts from first principles following a process of structured design.

Although the user definable properties that a user can set in this environment are persistent within the particular container, this persistence is destroyed the moment the object is moved to a different container environment. To overcome this problem two object methods *PersistDesignOut* and *PersistDesignIn* are introduced that will write the design data into an XML computer file on a local disk or an ftp directory on a remote project server. In this way structured design functions can be freely exchanged. As indicated in Figure 52 there is a bi-directional exchange of persistent data.

[D1] and [C1] are fictitious remote data sites that can be nominated by means of a data address within the object. This is achieved by setting the object property *DataLocation* to a valid URL. By means of the method *DisplayRemoteData* or by pressing the command button, these data will be displayed. The designer can then select the record and apply it to the current design object. Note that the data flow from remote data sources is uni-directional at this stage.

[1] There is a difference between easily available and surface features. Surface features make good indices to the extent that they are predictive of something important or useful.

The connection to the local database is conveniently achieved by means of the Microsoft ActiveX Data Object.

## 6.3 World Wide Web Implementation

Microsoft Internet Explorer supports the ActiveX controls. When the ARGOS design object is inserted into a web page the code looks like in Code fragment 4. The object starts with the label

```
<object classid="clsid:59DF65DF-632C-11D3-8D31-4854E8284FB0"
id="UserControl11" width="250" height="467">
```

and ends with the label

```
</object>
```

The `classid` is particularly important, because it is a totally unique code that is used to identify the particular class of the ActiveX control. This code is guaranteed to be unique in the world. This object was labelled with this code during the design and programming of the object. The `id` is the name that will appear on the list of possible controls when a user wants to insert an ActiveX control into his container software. In this case the `id` is `UserControl1`. The properties available for the object is exposed with the statements that read `<param name="_ExtentX" value="5292">`. In this example the ARGOS object contains 15 user definable properties. These properties fall into two main groups:

- Explicit
- Tacit

The explicit attributes have the prefix *AE_* and the tacit ones *AT_*. The explicit properties contain surface features (in CBR terminology) such as *gross area*, *net area*, *rentable area*, *construction area*, *volume*, *shape*, *durability*, *energy use* and *cost*. The tacit properties contain factors that were identified in Chapter 3, 3.5 where Kansei engineering was discussed in detail. This gives an indication of the sensory aspects of the design such as *sight*, *hearing*, *taste*, *smell*, *internal sensitivity* and *recognition*. Architecture has lot to do with the sensory aspects such as feeling of space, colour and acoustics.

It is apparent from the code fragment that the design detail is hidden away from the designer at this stage. The directly available properties make it possible to do basic preliminary feasibility studies. To make the detail visible the user will have to press the command buttons for CAD or Function that will unload the CAD drawing or the XML function tree. The ARGOS object also contains two buttons that a user can use to maximize or minimize the object. If a user wants to perform a specialised task he can invoke one of several object *methods* available.

```
<html>

<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<meta name="Template"
content="C:\PROGRAM FILES\MICROSOFT OFFICE\OFFICE\html.dot">
<meta name="GENERATOR" content="Microsoft FrontPage 4.0">
<title>AEDES</title>
</head>
```

```
<body background="aedes_b.gif" link="#0000FF" vlink="#800080"
bgproperties="fixed">

. . .

<object classid="clsid:59DF65DF-632C-11D3-8D31-4854E8284FB0" id="UserControl11"
width="250" height="467">
      <param name="_ExtentX" value="5292">
      <param name="_ExtentY" value="9885">
      <param name="BackColor" value="0">
      <param name="ForeColor" value="0">
      <param name="Enabled" value="0">
      <param name="BackStyle" value="0">
      <param name="BorderStyle" value="0">
      <param name="AE_grossarea" value="0">
      <param name="AE_nettarea" value="0">
      <param name="AE_rentable_area" value="0">
      <param name="AE_construction_area" value="0">
      <param name="AE_volume" value="0">
      <param name="AE_shape" value="0">
      <param name="AE_durability" value="0">
      <param name="AE_energy_use" value="0">
      <param name="AE_cost" value="0">
      <param name="AT_sight" value="0">
      <param name="AT_hearing" value="0">
      <param name="AT_taste" value="0">
      <param name="AT_smell" value="0">
      <param name="AT_internal_sensitivity" value="0">
      <param name="AT_recognition" value="0">
      <param name="AF_function" value="0">
</object>
. . .

<hr size="1" noshade color="#0000FF">

<p align="left"><font color="#0000FF" size="2" face="Arial"><!--webbot
bot="Timestamp" startspan s-type="EDITED"
s-format="%d %B %Y %I:%M %p" -->12 February 2000 06:46 AM<!--webbot bot="Timestamp"
i-CheckSum="54291" endspan --></font></p>
</body>
</html>
```

Code Fragment 4: ARGOS object placed in a web page (Author)

## 6.4 Hypothetical use of ARGOS

Due to time and financial constraints it is not the intention to develop a full commercial system in this study. However this section provides a run-through of how a designer might use the system.

It is assumed that a designer wants to design a new 16-bed male/female/paediatric in-patients section. The designer decides to see whether a previous conceptual layout of this type of facility exists. Unfortunately nothing exists in the office and a search of the web is also unsuccessful. It is also assumed that the ARGOS ActiveX control set is installed and available on the design workstation.

The designer decides that he will be using his Microsoft Excel Spreadsheet as a blackboard, because this is convenient for the type of design testing that he wants to do. The design brief specifies a design of not more than 260 m² and the cost should be below R 780 000-00. The accommodation requirement for the design is the following:

Capacity of 16 beds
Staff WC
Patient ablution

Sit bath
Nurse station
Duty room
Clean linen storage
Clean utility room
Ward kitchen
Dirty utility room
Store

The client states that it is a specific requirement that energy be saved especially with regards air conditioning.

The designer starts the process by calling up an Excel Spreadsheet with a default ARGOS control panel. He selects a minimal parametric ARGOS control from the Control Toolbox (ARGOS.CBR) and inserts it into the spreadsheet (Figure 53).

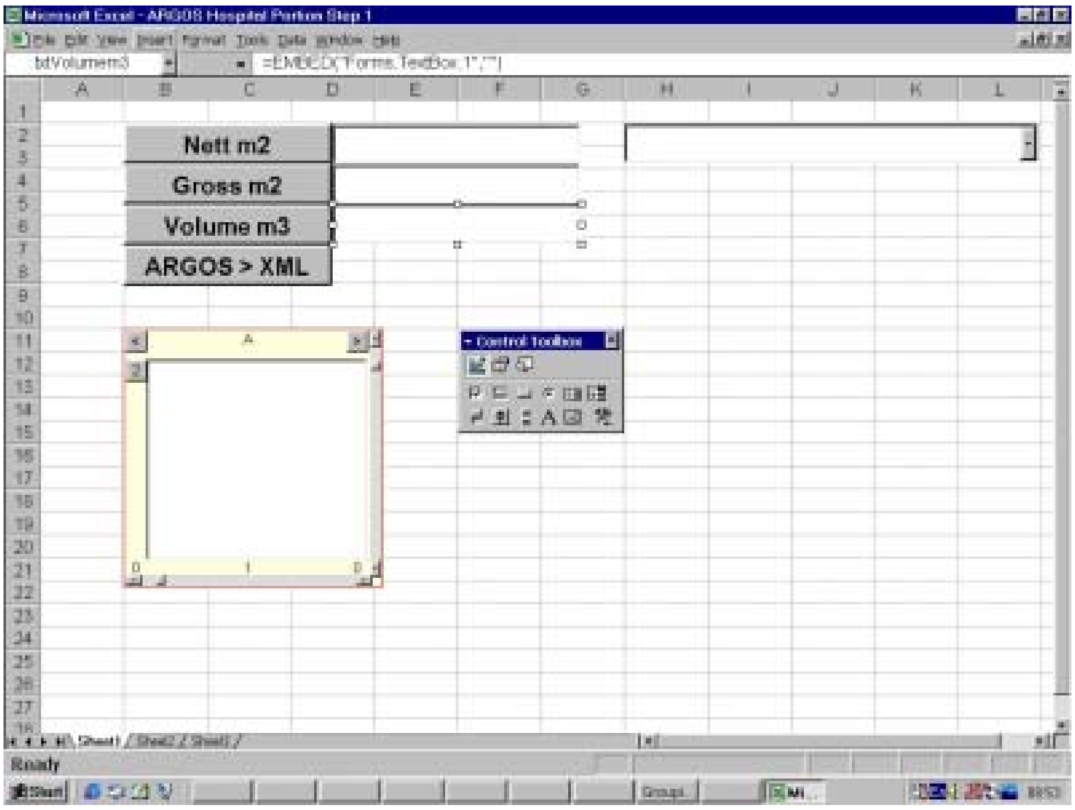


Figure 8: Design of 16 bed male/ female/ paediatric in-patients section step 1 (Author)

At this stage the spreadsheet contains four command buttons that enable the designer to calculate net m², gross m², volume m³ and a special button that enables him to export the design in XML format to an XML aware CAD system for subsequent detailed design. For convenience a combobox is also included where the designer can list the spaces in the design. At the moment the ARGOS control is still default size and the internal properties all have default or undefined values.

The designer now continues to develop the design according to the brief and his experience. After some time the design looks like in Figure 54. The design contains 14 ActiveX controls of varying size. The designer adjusts some of the properties that will be important for subsequent retrieval of previous design cases. He sets the wall thickness in the *M_wall1*, *M_wall2*, *M_wall3* and *M_wall4* properties to respectively 55, 220, 55 and 220. Seeing that this is a special section intended for paediatrics some Kansei adjectives are added in the *AT_hearing*, *AT_internal_sensitivity*, *AT_recognition*, *AT_sight*, *AT_smell* and *AT_taste* properties.

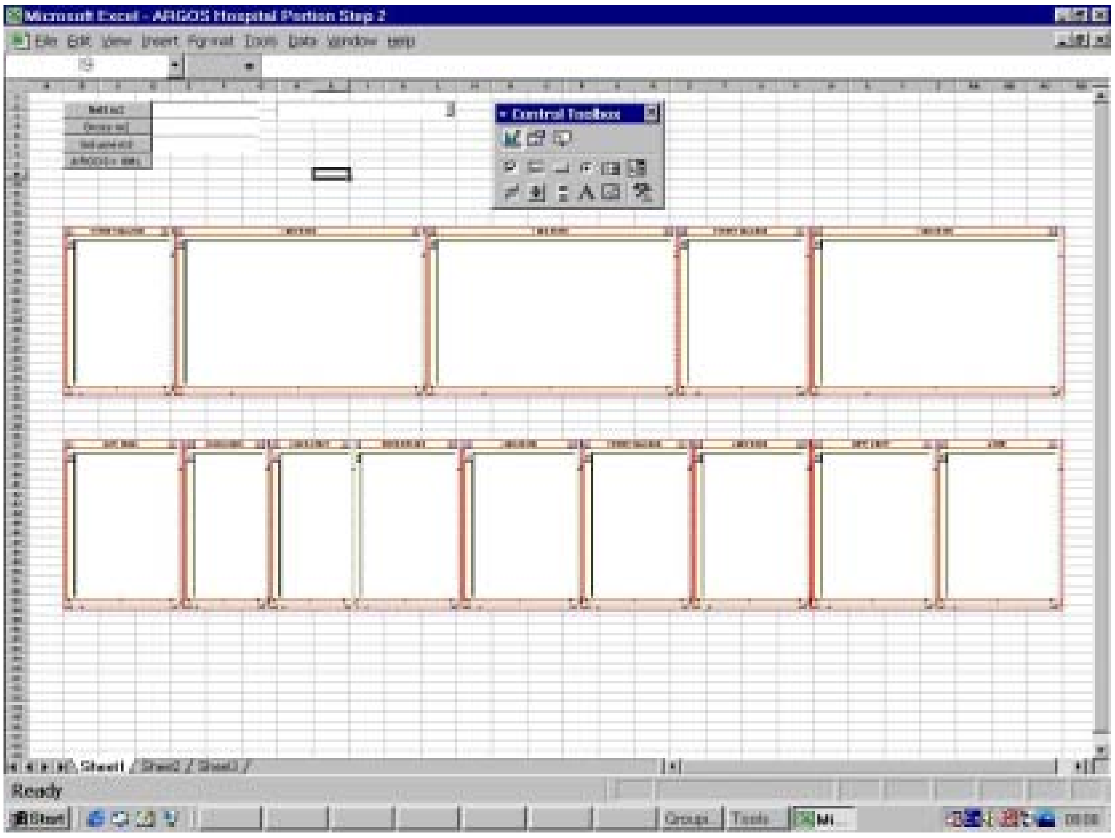

Microsoft Excel - ARGOS Hospital Portion Step 2
File Edit View Insert Format Tools Data Window Help
Control Toolbox
Sheet1 / Sheet2 / Sheet3 /
Ready
Start

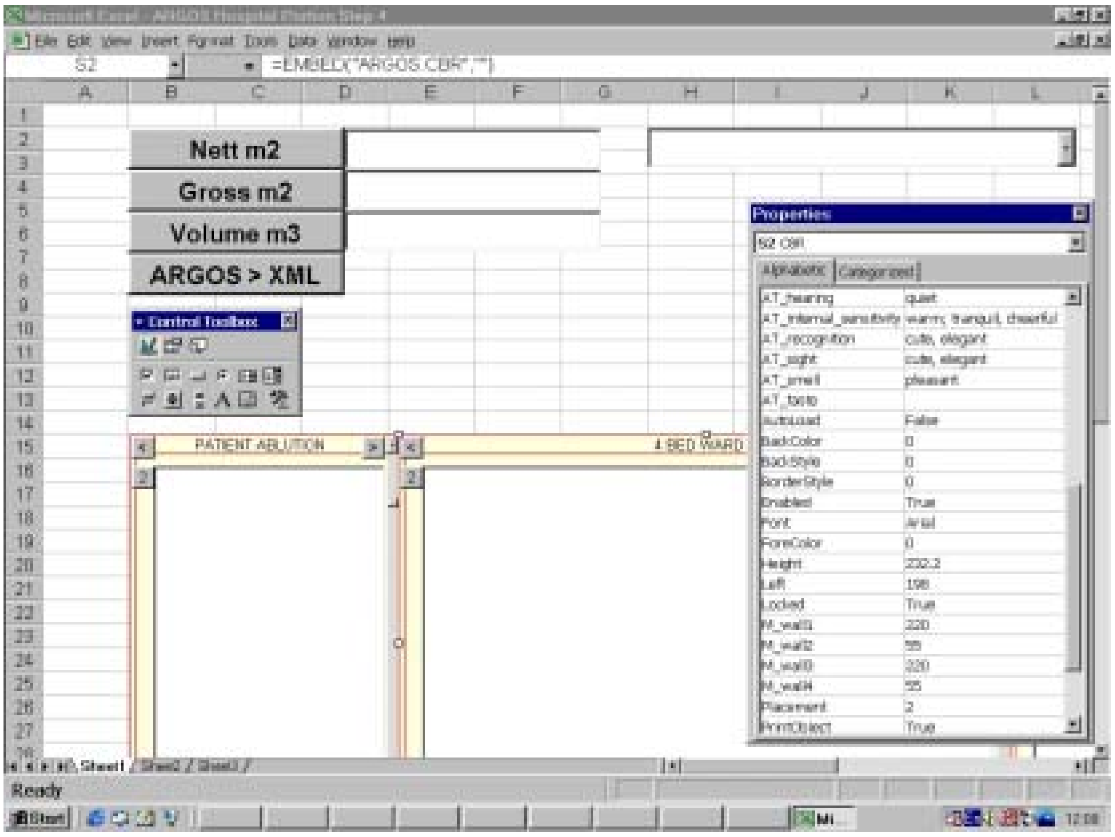
Figure 9: Design of 16 bed male/ female/ paediatric in-patients section step 2 (Author)

Some useful adjectives that could be used are listed in Figure 27. The Kansei properties now read:

*AT_hearing = quiet*
*AT_internal_sensitivity = warm, tranquil, cheerful*
*AT_recognition = cute, elegant*
*AT_sight = cute, elegant*
*AT_smell = pleasant*
*AT_taste =*

These properties are very important for subsequent retrieval of possible previous design experience. The ARGOS properties can be defined by typing directly into the relevant property, set by program or indirectly adjusted by means of the *x*, *y* or *z* slide controls. The properties are also useful because they can be directly transferred to CAD systems that support the definition of attributes such as MicroGDS or AutoCAD.

At this stage a typical property list for the 4 Bed Ward would look like the one illustrated in Figure 55. Note the Kansei definitions at the top of the list and the various wall thickness properties at the bottom of the list.



S2 CBR

| | |
|---|---|
| AT_hearing | quiet |
| AT_internal_sensitivity | warm, tranquil, cheerful |
| AT_recognition | cute, elegant |
| AT_sight | cute, elegant |
| AT_smell | pleasant |
| AT_taste | |
| AutoLoad | False |
| BackColor | 0 |
| BackStyle | 0 |
| BorderStyle | 0 |
| Enabled | True |
| Font | Arial |
| ForeColor | 0 |
| Height | 232.2 |
| Left | 198 |
| Locked | True |
| M_wall1 | 220 |
| M_wall2 | 55 |
| M_wall3 | 220 |
| M_wall4 | 55 |
| Placement | 2 |
| PrintObject | True |

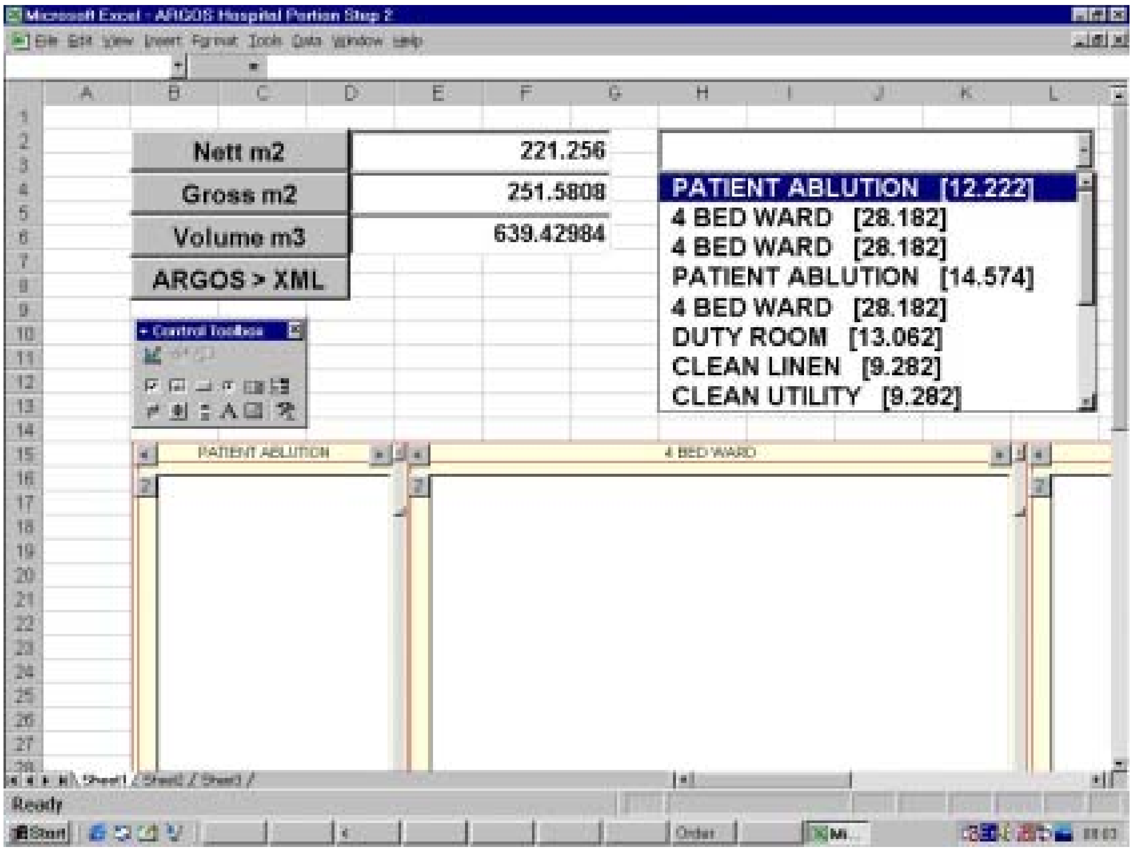
Figure 10: Setting design properties of a paediatric ward (Author)

At this stage the designer would like to know the gross and net area. This is accomplished by selecting the relevant command buttons that start Visual Basic routines that scan through all the design controls present and retrieve all the areas. In a similar way the volume is determined.

The system reports the following (Figure 56):

Net m$^2$ = 221.256
Gross m$^2$ = 251.5808
Volume m$^3$ = 639.42984

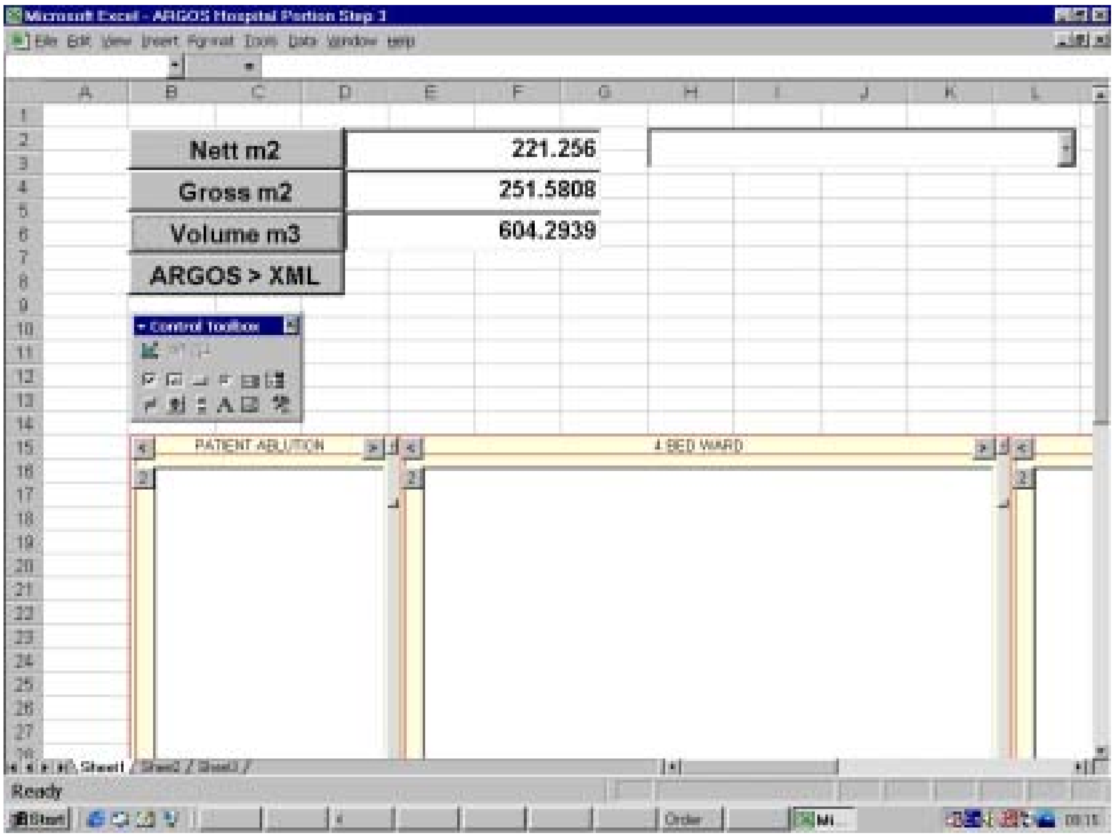
At this stage he is not sure what the construction cost per m$^2$ is and activate his Microsoft Internet Explorer. He accesses the http://design.case.co.za web page that is one of the available design information sites to search for an estimated construction cost/m$^2$ for this type of facility. This is conceptually illustrated in Figure 52. He finds this cost to be R 2800-00/m$^2$. On the basis of this it is estimated that it would cost R 704 426-24 to build this facility. In the meantime the air conditioning engineer is analysing the design from an energy point of view. He finds that the current volume would require a significantly larger installation than originally anticipated. In an attempt to solve this the ceiling height is lowered from 2 890mm (34 brick courses) to 2 720mm (32 brick courses) (Figure 57). This is accomplished by setting the *AA_zdim* property to 2 720 for each ARGOS component. The recalculation indicates that the volume has now in fact been reduced from 639.43 m$^3$ to 604.29 m$^3$. This is an immediate saving of 5,5%. In a similar way other direct design parameters can be adjusted and tested.


Microsoft Excel - ARGOS Hospital Portion Step 2
File Edit View Insert Format Tools Data Window Help
Control Toolbox
PATIENT ABLUTION
4 BED WARD
Sheet1 / Sheet2 / Sheet3 /
Ready
Start

| | |
|---|---|
| Nett m2 | 221.256 |
| Gross m2 | 251.5808 |
| Volume m3 | 639.42984 |
| ARGOS > XML | |

PATIENT ABLUTION [12.222]
4 BED WARD [28.182]
4 BED WARD [28.182]
PATIENT ABLUTION [14.574]
4 BED WARD [28.182]
DUTY ROOM [13.062]
CLEAN LINEN [9.282]
CLEAN UTILITY [9.282]

Figure 11: The calculation of area and volume (Author)


Microsoft Excel - ARGOS Hospital Portion Step 3
File Edit View Insert Format Tools Data Window Help
Control Toolbox
PATIENT ABLUTION
4 BED WARD
Sheet1 / Sheet2 / Sheet3 /
Ready
Start

| | |
|---|---|
| Nett m2 | 221.256 |
| Gross m2 | 251.5808 |
| Volume m3 | 604.2939 |
| ARGOS > XML | |

Figure 12: The reduction of volume by lowering the ceiling (Author)

The designer is now satisfied with the basic design and continues with the detailed design. By double clicking[1] on the ward the search engine is invoked to search for previous design cases that fit the type and dimensions previously captured. Initially only the main description is used to search for a list. The system reports that three types of ward is available:

Two Bed Ward
Four Bed Ward
Observation/ Trauma Ward

He selects the Four Bed Ward. ARGOS now uses the *AA_xdim*, *AA_ydim*, *AA_zdim* as well as the set of Kansei descriptions such as *AT_hearing*, *AT_internal_sensitivity*, *AT_recognition*, *AT_sight*, *AT_smell* and *AT_taste* as search parameters. This is implemented with the dynamic linguistic variable method described in detail in Chapter 3. Only one solution is found and placed into the design (Figure 58). In a similar way other parts of the design can be developed and further refined. Once the designer is satisfied he can export the entire design to an XML aware CAD system or a rendering/ visualisation package for detailed design and the production of working drawings.

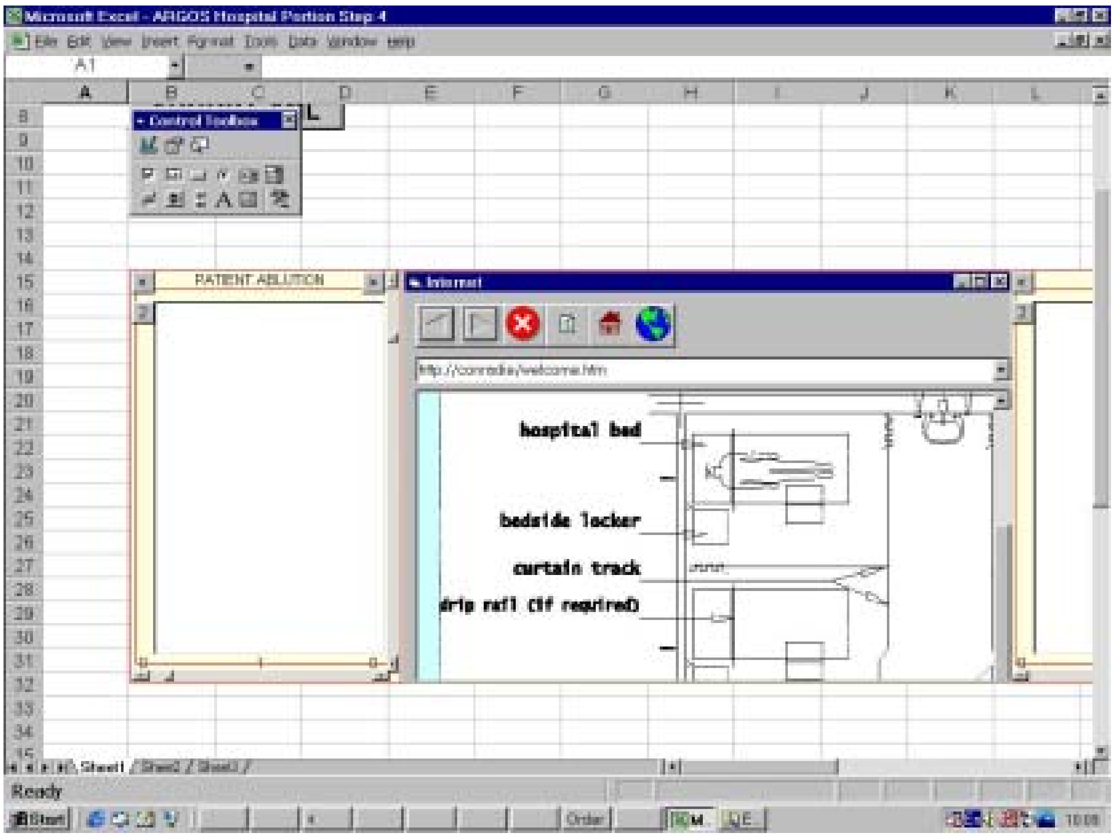


Figure 13: The retrieval and insertion of a Four Bed Ward detailed case (Author)

## 6.5 Empirical response tests

To ensure that the proposed system would be scalable and could eventually be applied to real problems in the architectural design domain a series of response tests were conducted. A recently completed very large shopping centre analysed to establish the needs of the

[1] Two possibilities exist to implement the CBR retrieval in ARGOS. The basic ARGOS can switch to CBR mode or a special separate ARGOS control can be written to handle only this aspect. The final implementation will become clearer with continued research.

professional team consists of 57 spaces on the lower first floor, 148 on the ground floor and 73 on the upper first floor. The proposed component system should be capable of supporting the following types of design activities in large and complex designs:

- Concept selection
- Retrieval of design experience
- Test of spatial relationships
- Scenario planning
- Collaboration on a global basis
- Modelling and simulation

To support and implement these activities require the ARGOS components to support any combination of parametric, *Rule-based*, *Model-based* and *Case-Based* methodologies. The tests concentrated on how responsive the components are to return direct and derived parametric values such as gross area and wall-space ratio. The prototype component presently supports 30 primary design properties, inter linked where appropriate.

The efficient response is primarily due to the fact that the parametric calculations are performed inside the ARGOS components where it is optimal and in a compiled form. The software that interrogated the components was, in this case, Visual Basic for Applications running at a more moderate interpreted speed than compiled Visual Basic. Although the prime purpose of the system is not efficient response, but rather opportunistic control, flexibility and interfacing to external software these tasks need to be accomplished within reasonable time.

The tests were conducted on a Microsoft Excel spreadsheet used as a blackboard, because it is so widely used and offers a convenient interface to spreadsheet capabilities and analysis software. It is clear that the slow computer (266 MHz CPU with 32 MiB[1] of RAM) is efficient up to about 75 components, whereas the moderate and fast computers are still efficient well beyond 100 components. The tests consisted of a Visual Basic program requesting parametric values that could be used in complex external analysis programs (Figure 59).

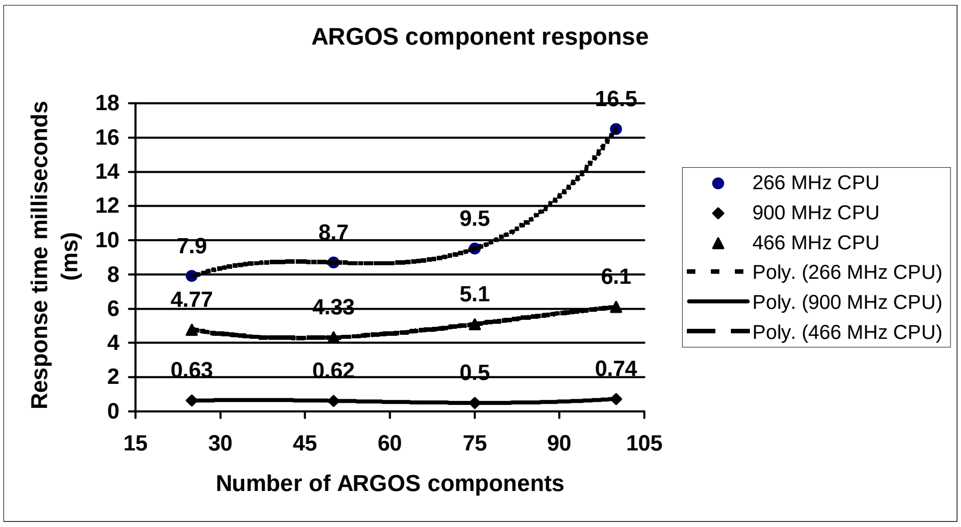


Figure 14: ARGOS component response (Author)

[1] One mebibyte (MiB) is equivalent to 1 048 576 bytes whereas one megabyte is equivalent to 1 000 000 bytes.

The increase in file size is linear (Figure 60). It should be noted that the size is expressed in kibibytes[2] as recommended by the International Electrotechnical Commission for binary multiples in December 1998.

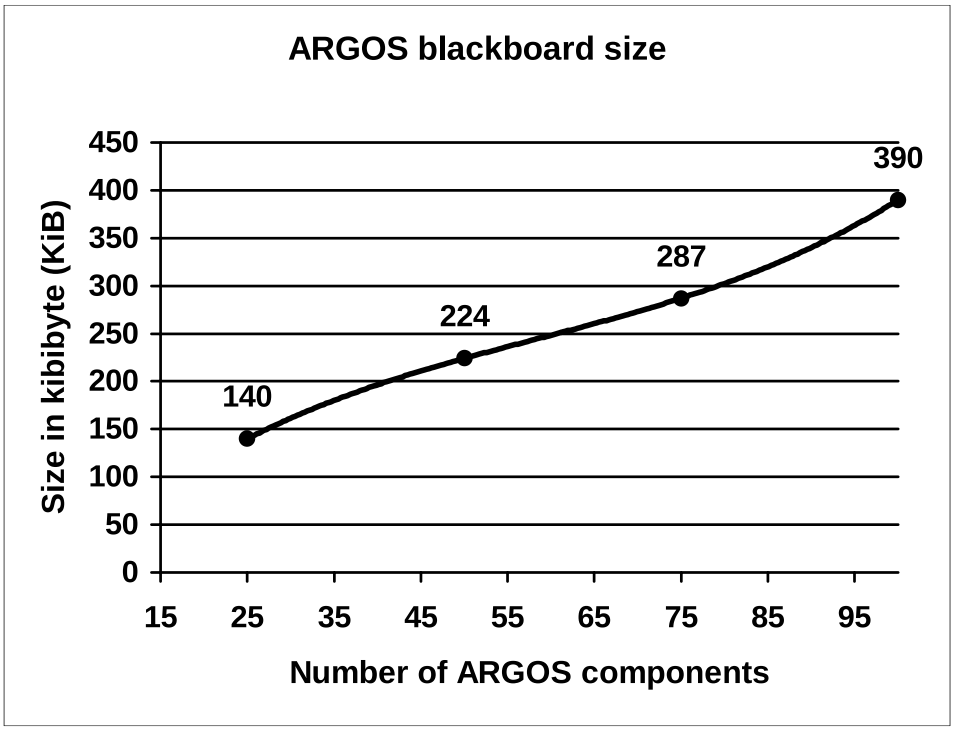


Figure 15: ARGOS blackboard size (Author)

## Summary

A life cycle information infrastructure based on XML is used as a basis for ARGOS. The use of XML as a design language facilitates design knowledge delivery to users. The use of Cascading Style Sheets, XSL and VML was explored. This proves the versatility of XML beyond doubt. The storage of a CAD drawing in XML was analysed in detail.

Due to the generic nature of ARGOS the range of possible applications is large. The role in structured planning and design knowledge delivery is proposed. The relationships of ARGOS to other intelligent data sources were explored.

A detailed parametric ARGOS object was written with the ability to switch between 2D and 3D modes. A compact miniature Internet browser was developed that could be combined with the basic ARGOS component. This will enable unlimited data access.

Internally the ARGOS design case should contain four main types of design information consisting of both alphanumeric and graphic information:

- Tacit design information
- Explicit design information
- Graphic information in the form of XML
- Functional design information and constraints

[2] One kibibyte (KiB) is equivalent to 1 024 bytes whereas one kilobyte (kB) is equivalent to 1 000 bytes.

Finally a short hypothetical run-through of how the ARGOS system might be used is described. Empirical response tests were also conducted for a blackboard (spreadsheet) with 25, 50 and 100 controls on different types of computer. This indicates that the ARGOS blackboard type of architecture using a spreadsheet is effective.

# Chapter 7: Summary, Conclusions, Recommendations and Assessment

## Introduction

This chapter concludes the study with a summary of the research work, conclusions, recommendations for further work and a critical assessment of what has been achieved.

## 7.1 Summary

### 7.1.1 Out-of-industry methodologies

Product innovation methodologies are useful at various tacit and explicit levels. AI can play an important role in Knowledge Based Design. CBR is a promising sub-field of AI that can greatly contribute to the contextual storing of design knowledge. It is clear that AI should be used more in the background and especially in architecture automatic adaptation of designs should not be attempted. CBR, RBR and MBR should be not be seen in isolation but should rather be viewed as a continuum of techniques.

PREMIS provides useful insights into a deeper understanding of ontology in a Facilities Management environment. The development of PREMIS highlighted the inadequacies of Relational Databases with regards hierarchical structures. The problems with ontology will be perpetuated in the BPMs and life cycle information infrastructures investigated in this study.

Knowledge Management is becoming very prominent although there are still unsolved problems. However many researchers are working on the particular sub-problems due to the importance of this for the global economy. *Concept extraction* and *Natural Language Processing* remains problematic, however significant progress has already been made.

Fuzzy sets are useful for aiding the retrieval of design knowledge in general and cases specifically by means of dynamic linguistic variables. The semantic differential method of Snider and Osgood (Snider *et al*. 1957) and the semantic differential adjectives as used by Nagamachi bear a relationship to the approach advocated by the author.

The analysis of the characteristics of manufacturing such as process, flow and throughput indicate that these are not directly applicable to the problem under consideration, but should rather be applied at the process level. The theories of Goldratt indicate that the manufacturing environment is particularly applicable for Theory of Constraints and that the system optimum is not the sum of the local optima. Concurrent Engineering is an important technique to avoid the so-called time-trap. This is where the life cycle time of products are decreased while the time spent on product development is greatly increased.

Taguchi techniques indicate the importance of off-line and on-line quality control. These indicate that quality is related to the loss to society caused by a product during its life cycle. In terms of the current thesis these methods should rather be used to select appropriate materials to minimise life cycle costs in the context of sustainable development.

Kansei Engineering (KE) is a mature and useful technique to quantify cognition and product image in such a way as to influence the product development process. KE operates at a very high tacit level and could make a significant contribution to the storage of tacit architectural design information.

The QFD exercise undertaken indicates that the ability to generate what-if scenarios across the project life cycle as the most important user requirement. QFD as a technique to extract raw architectural user requirements was pioneered in the AEDES system. QFD is useful if the time and cost can be justified. QFD is an important technique in the manufacturing industry and was one of the techniques that saved the Detroit automotive industry from ruin in the face of severe competition from Japan. The contribution that QFD can make in architecture is dependent on the general acceptance of this slightly elaborate technique.

TRIZ is a powerful method to solve inventive problems. However the present available commercial TRIZ software emphasise engineering type of problems. A significant amount of work will have to be done to make its use tractable in architectural design.

### 7.1.2 Life cycle design knowledge

The use of objects is the preferred way to achieve abstraction, generalisation and interaction in systems supporting the life cycle development process. The unique way that objects are used in the precedent systems PREMIS and AEDES provides valuable insights. Microsoft ActiveX controls are a convenient means to implement ARGOS.

A life cycle information infrastructure based on XML is used as the basis for ARGOS. The use of XML as a design language facilitates design knowledge delivery to users. The use of Cascading Style Sheets, XSL and VML is explored. This proved the versatility of XML beyond doubt. The storage of a CAD drawing in XML is analysed in detail.

### 7.1.3 ARGOS intelligent component

Although AEDES can be viewed as a failure in commercial terms, it provided the first insights into the how user requirements might be extracted and structured to obtain a performance requirement. It was discovered that requirements and functions fall into two main groups i.e. active and passive. A set of functions to successfully enable the operational capability of the requirement could be identified. These functions could be individually characterised according to functional and physical characteristics and constraints. Finally a characterised function could be allocated to a physical element. This transformational method should however not be overemphasised.

An intelligent design component should be both object based and Internet enabled. Its knowledge must be structured and self-documenting. It should be able to operate in a wide variety of environments over a long period of time. It must be useful in small and large project teams using different design and construction processes. It cannot be predicted with certainty what the nature of these numerous processes will be in future.

ARGOS could improve activities such as:

- Concept selection
- Retrieval of design experience
- Test of relationships
- Scenario planning
- Collaboration on a global basis
- Modelling and simulation

Due to the generic nature of ARGOS the range of possible applications is large. The new innovative role in structured planning and design knowledge delivery is proposed. The relationships of ARGOS to other intelligent data sources were also explored.

A detailed parametric ARGOS object was written with the ability to switch between 2D and 3D modes. A compact miniature Internet browser was developed that could be combined with the basic ARGOS component. This will enable unlimited data access.

Internally the ARGOS design case should contain four main types of design information that consists of both alphanumeric and graphic information:

- Tacit design information
- Explicit design information
- Graphic information in the form of XML
- Functional design information and constraints

## 7.2 Conclusions

The focus of this study was to discover how useful techniques from the world of manufacturing and Artificial Intelligence are in the light of the unique characteristics of the early phases of design. To achieve these three sub-problems were identified.

In *sub-problem 1*, the hypothesis was that a building can be seen as a production product and hence established Systems Engineering techniques and quality measures can be applied to the briefing and design process. Although there are many similarities between the two industries in the sense that efficiency and globalisation are forcing improvement there are also significant and problematic differences. The numerous examples studied from the various out-of-industry perspectives indicated that structured methodologies can only assist at the various tacit and explicit levels, never equal the phenomenal creative capabilities of the human brain. It is clear that the techniques should not be seen in isolation, but rather forms a continuum that should be used when appropriate. Rule-Based Reasoning is useful with for example acoustical design, Model-Based Reasoning assists with the testing of structural behaviour of a design, wind loads and circulation. Case-Based Reasoning is useful to remember previous designs and as such extend the designer's vocabulary. Due to the importance of context in architectural design CBR proved to be useful as a means of recording past experience for possible future use and in the process expedite the design process.

In *sub-problem 2*, the hypothesis was that the architectural briefing and design process could be structured in such a way that it can be implemented on a software system to ensure total life cycle design. The surprising discovery was made that the structuring and storage of in-context design information will become more important than the software application that originally created it. This study has shown that the use of XML as a design language is a viable alternative for the highly complex Building Product Models currently proposed. The flexible structure and extendibility of XML forms a useful basis for a design language and further processing can be achieved by means of style languages such as CSS and XSL. It is already viable to display XML data in vector format by means of VML and the SVG standard announced very recently. The other forthcoming W3C standards such as the XML Fragment Interchange Working Draft and XML Query Data Model will leverage XML even further. Although XML was not primarily intended to be used as a means of storing BPM type of knowledge the inherent characteristics makes it very powerful. According to Cohen (2000:257) XML will, if it widely adopted, result in data sharing and electronic commerce in the building industry on a scale not previously imagined. This is confirmed by the recent (September 1999) first meeting of the aecXML Working Group that was held in Dallas, Texas, attended by over 130 companies and organisations (Cohen 2000:259).

The intelligent component ARGOS, together with the miniature Internet browser developed, makes it feasible to source any design information (graphic and alphanumeric) from anywhere in the world inside any number of autonomous instances of ARGOS in any Microsoft compliant container. ARGOS is a non-prescriptive autonomous component that makes it possible to use any external process to manipulate or interrogate it. ARGOS can be used at any level of specificity, but practical considerations are likely to limit it to larger architectural units such as spaces, buildings and facilities. ARGOS succeeded in integrating the two important technologies of object-oriented design and XML.

It is clear that the approaches previously followed in software products placed the software application more centrally. This is seen in movements such as Knowledge-Intensive CAD. Microsoft started a new approach that placed the intelligent electronic document in a more central position. The possibilities of this approach were explored throughout the dissertation. Although this is very different from the more traditional approach followed in PREMIS and the prototype AEDES, it is more likely to succeed. The reasons for this can be summarised as:

- A Low-level minimum entry-level platform can be used
- Low cost starting point
- The convenience of portable object technology possibly structured in the form of cases
- Avoids the use of expensive CAD during the early briefing and design stages
- The same technology can be used in small and very complex applications
- The starter kit (intelligent design broker) can be used throughout the life cycle of the building
- Independent third parties can produce starter kits (packaged cases)
- The user can use his own case templates to facilitate what if questions
- Ubiquitous availability of structured design knowledge through the Internet for international teams

In *sub-problem 3*, the hypothesis was that architectural designs and design parameters can be quantified and electronically packaged in such a way as to expedite future designs that require similar designs or parts of designs. The Microsoft object technologies (ActiveX) as well as the modern interpretation of Object Oriented Computer Aided Design prove that it is the preferred way to store architectural designs. If the power of the Internet is taken account of then the design language that is likely to be preferred would be XML.

## 7.3 Recommendations for further work

The ARGOS methodology would still require more research to refine the initial prototype. It is envisaged that different generic types of ARGOS objects should be created to support the various aspects of the design activities better. The problem of adaptation was not adequately addressed in this thesis. The implementation of constraints will also need more research.

From the work done in AEDES it is clear that an important requirement is the further development of technologies such as Construction Project Simulators. This could be based on ARGOS type technology to provide professionals with the ability to manipulate project information across the life cycle of a facility and to create what-if scenarios by means of affordable desktop tools.

A major challenge is to change to a paradigm where the life cycle information infrastructure becomes primary and the applications that use it secondary. The current international emphasis of XML indicates that it is important that researchers first establish a sound BPM (the means of expressing design information) basis before placing too much emphasis on formalising processes.

## 7.4 Assessment

Although exciting breakthroughs were made in this study with the discovery of an intelligent component that could be used to bridge structured methodologies and the creative aspects of design there are unfortunately also a number of shortcomings such as:

- XML was never designed as a means to structure design knowledge. It will take a while to formalise and prove the idea to other academics and the commercial world.
- The graphic XML examples shown still have mistakes in the processing of text and the correct translation of the bulge factors for arcs and circles. At this stage the XSL style sheet translation into VML is still primitive and slow. This will improve significantly as SVG begins to gain wider acceptance. However the success of XML is fortunately not dependent on these technologies that use it as basis for processing. With the information overload experienced in the global community the ability to deliver information profiled to the needs of the individual knowledge worker is important. Here XML is again a useful means of achieving this.
- ARGOS in its present form is very orthogonal and makes the design of organic free flowing forms difficult. It is impossible to say whether designers will accept this new type of blackboard autonomous component approach in design. At the moment CAD still dominates the production of drawings although experience has shown that the contribution of CAD to the creative aspects of design has been disappointing over many years.
- The Internet bandwidth in South Africa is at the moment too slow for the collaborative projects envisaged.
- The unsatisfactory contribution up to date of Knowledge-Based Computer-Aided Architectural Design in the conceptual stages of design will make critics highly sceptic as to the possibilities of ARGOS. In the light of the exciting new information supporting technologies available today, the author is of the opinion that a sensible solution is imminent.

# References

Agility Forum. 1997. *Next-generation manufacturing*. Bethlehem, PA : Agility Forum.

AKAO, Y. 1997. QFD: Past, present and future, in *Proceedings of the third annual international QFD symposium*. Linköping, Sweden.

ANUMBA, C.J., Baron, G. & Evbuomwan, N.F.O. 1997. Communications issues in concurrent life-cycle design and construction, in *BT Technology Journal,* 15(1), Jan. 1997.

APPLEMAN, D. 1999. *Developing COM/ ActiveX components with Visual Basic 6*. Indianapolis, IN. : Sams.

BELLMAN, R. 1978. *An introduction to artificial intelligence: can computers think?* San Francisco, California : Boyd & Fraser Publishing Company.

BELLMAN, R.E. & Zadeh, L.A. 1970. Decision-making in a fuzzy environment, in *Management Science,* 17(4), Dec. 1970:141-164.

BERNDES, S. & Stanke, A. 1996. A concept for revitalisation of product development, in *Concurrent simultaneous engineering systems*. Berlin : Springer-Verlag.

BIEMEL, W. 1976. *Martin Heidegger. An Illustrated Study by Walter Biemel*. London, New York: Harcourt Brace Jovanovich.

BOHN, R.E. 1994. *Measuring and managing technological knowledge*. Sloan Management Review Association.

BOJADZIEV, G. & Bojadziev, M. 1995. *Fuzzy sets, fuzzy logic, applications*. Singapore : World Scientific.

BöRNER, K. 1998. CBR for design, in *Lecture Notes in artificial intelligence: Case-based reasoning technology*. Berlin : Springer-Verlag.

BOUZEGHOUB, M., Gardarin, G., & Valduriez, P. 1997. *Object Technology: Concepts and Methods*. Boston : International Thomson Computer Press.

CARRARA, G. & Kalay, Y.E. 1994. *Knowledge-based computer-aided architectural design*. Amsterdam : Elsevier.

CHARLTON, C.T., Ball, N.R. & Matthews, P.C. 1998. Towards mechanical design object reuse, in *Artificial intelligence in design.* Dordrecht: Kluwer Academic Publishers.

COAD, P. & Yourdan, E. *Object-Oriented Analysis*. Englewood Cliffs, NJ: Yourdan press.

COHEN, L. 1995. *Quality Function Deployment: How to make QFD work for you*. Reading, MA : Addison-Wesley.

COHEN, J. 2000. *Communication and Design with the internet.* New York : W.W. Norton & Company.

CONRADIE, D.C.U. & Küsel, K. 1999. The use of QFD for architectural briefing and design, in *Transactions from the eleventh symposium on Quality Function Deployment*. Novi, Michigan : QFD Institute.

CONRADIE, D.C.U. & Abbott, G. 1996. The use of a Facilities Management System for the South African National Health Facilities Audit, in *Proceedings of the CIB W70 Symposium. Helsinki*:71-74.

CRAIG, J.C. & Webb, J. 1997. *Microsoft visual Basic 5.0 developer's workshop.* 4th edition. Redmond, Washington : Microsoft Press.

DAVENPORT, T.H. 1997. *Information Ecology: Mastering the information and knowledge environment*. New York, NY : Oxford University Press.

DEBENHAM, J.K. 1998. *Knowledge engineering*. Berlin : Springer-Verlag.

DREYFUS, H.L. 1993. *What computers still can't do*. Cambridge, Mass. : The MIT Press.

DETTMER, H.W. 1997. *Goldratt's theory of constraints: A systems approach to continuous improvement*. Milwaukee, Wisconsin : Quality Press.

DOMESHEK, E.A., Kolodner, J.L. & Zimring, C.M. 1994. The Design of a Tool Kit for Case-Based Design Aids. *Internet:* http://www.cc.gatech.edu/aimosaic/faculty/kolodner/muse.html. Accessed: 03 October 2000.

EASTMAN, C.M. 1999. *Building Product Models: Computer Environments Supporting Design and Construction*. Boca Raton : CRC Press.

ENGELBRECHT, B. 1998. Business Engineering: the object oriented framework, in *Course documentation, DISCON Specialists CC*. CSIR Training Centre, Pretoria.

FEIJO, B., Rodacki Gomes, P.C., Bento, J., Scheer, S. & Cerqueira, R. 1998. Distributed agents supporting event-driven design processes, in *Artificial intelligence in design*. Dordrecht : Kluwer Academic Publishers.

FLEMMING, U. 1994. Artificial Intelligence and Design: A Mid-term Review, in *Knowledge-based computer-aided architectural design*. Amsterdam : Elsevier.

FLEMMING, U. 1994. Case-based design in the SEED system, in *Knowledge-based computer-aided architectural design*. Amsterdam : Elsevier.

FOX, S. 1995. *Introspective Learning for Case-Based Planning*. PhD thesis, Indiana : Department of Computer Science, Indiana University.

GartnerGroup. 1998. *Knowledge management innovation*. Stamford, Connecticut : GartnerGroup.

GartnerGroup. 1998. *Collaboration and groupware*. Stamford, Connecticut : GartnerGroup.

GartnerGroup. 1998. *Knowledge management architectures*. Stamford, Connecticut : GartnerGroup.

GOEL, A.K., Kolodner, J.L., Pearce, M. & Zimring, C. 1991. Towards a Case-Based Tool for Aiding Conceptual Design Problem Solving, in *Proceedings of the DARPA Case-Based*

*Reasoning Workshop*. Washington : Defense Advanced Research projects Agency Information Science and Technology Office.

GOLDRATT, M & Cox J. 1993. *The Goal*. Aldershot, Hampshire : Gower Publishing Company.

GOLDRATT, E. 1990. *What is this thing called theory of constraints*. Croton-on-Hudson, New York : North River Press.

GROBLER, L.J., Knoetze, T. & Truter, R. 1997. *BEARS Building Enivironmental Assessment and Rating System for South Africa*. Pretoria : CSIR Division of Building Technology.

GROOVER, M. 1996. *Fundamentals of modern manufacturing*. Upper Saddle River, New Jersey : Prentice Hall.

HARARI, O. 1999. *Leapfrogging the competition*. Rocklin, California : Prima Publishing.

HAROLD, E.R. 1999. *XML Bible*.  Foster City, California : IDG Books WorldWide.

HILL, R.C., Pienaar, J., Bowen, PA. &Küsel. 1998. The transition to sustainability in the planning, construction and management of the built environment in South Africa, in *Proceedings of the CIB World Building Congress*, 7-12 June, 1998, Gävle, Sweden.

HINRICHS, T.R. 1991. *Problem Solving In Open Worlds: A Case Study In Design*. PhD thesis, Atlanta : Artificial Intelligence Group College of Computing, Georgia Institute of Technology.

HJöRLAND, B. 1994. Nine principles of knowledge organization, in *Knowledge Organization and Quality Management*. Frankfurt/ Main: Indeks Verlag.

IEEE Computer Society. 1996. *IEEE Std 1233-1996: IEEE Guide for developing system requirements specifications*. New York, NY : Institute of Electrical and Electronics Engineers.

International Alliance for Interoperability. 1997. *Industry foundation classes*. Washington, DC : IAI.

KAPLAN, S. 1996. *An introduction to TRIZ: the Russian theory of inventive problem solving*. Ideation International, Inc.

KESSLER, S. 1996. Enabling technologies I, the CONSENS platform, in *Concurrent simultaneous engineering systems*. Berlin : Springer-Verlag.

KLIR, G.J. & Yuan, B. 1995. *Fuzzy sets and fuzzy logic: Theory and applications*. Upper Saddle River, NJ. : Prentice Hall.

KORN, G. 1996. MDS: A system for decision support in the economic efficiency analysis and controlling of the product developing process, in *Concurrent simultaneous engineering systems*. Berlin : Springer-Verlag.

KOLODNER, J. 1993. *Case-based reasoning*. San Mateo, California : Morgan Kaufmann Publishers.

KOLODNER, J.L. 1996. Making the implicit explicit: Clarifying the principles of Case-Based Reasoning, in *Case-Based Reasoning* edited by D.B. Leake. Menlo Park, California : AAAI Press.

KOLODNER, J.L. 1996. A tutorial introduction to CBR, in *Case-Based Reasoning* edited by D.B. Leake. Menlo Park, California : AAAI Press.

KüSEL, K. 2000. *The Requirements of an Integrated Project Environment in the South African Construction Industry*. MSc thesis, Pretoria : Faculty of Engineering, the Built Environment and Information Technology, University of Pretoria.

LASZLO, E. 1996. *The systems view of the world*. Cresskill, NJ. : Hampton Press.

LAWRENCE, J. 1993. *Introduction to Neural Networks*. Nevada City, CA.: California Scientific Software Press.

LEIBMANN, M. 1999. *A Way to KM Solutions. Things to Consider when Building Knowledge Management Solutions with Microsoft Technologies*. Microsoft.

LOMAX, P. 1997. *Laura Lemay's web workshop ActiveX and VBScript*. Indianapolis, Indiana : Sams.net Publishing.

MAZUR, G. 2001. Theory of Inventive Problem Solving (TRIZ). *Internet:* http://www-personal.engin.umich.edu/~gmazur/triz/ .Accessed: 26 June 2001.

MEADOWS, D.H., Meadows, D.L., Randers, J. and Behrens, W.W. 1975. *The limits to growth*. London : Pan Books.

MENGES, R. & Eigenmann, U. 1996. Design of production facilities, in *Concurrent simultaneous engineering systems*. Berlin : Springer-Verlag.

MEYER, B. 1988. *Object-oriented software construction*. New York : Prentice Hall.

NAGAMACHI, M. 1999a. Kansei Engineering and new product development, in *Kansei Engineering Workshop at the eleventh symposium on Quality Function Deployment*. Novi, Michigan : QFD Institute.

NAGAMACHI, M. 1999b. Kansei engineering and its applications in automotive design, in *Kansei Engineering Workshop at the eleventh symposium on Quality Function Deployment*. Novi, Michigan : QFD Institute.

NONAKA, I. 1998. The knowledge-creating company, in *Harvard business review on knowledge management*. Boston, Mass. : Harvard Business School Publishing.

O'SULLIVAN, B. & Bowen, J. 1998. A constraint-based approach to supporting conceptual design, in *Artificial intelligence in design*. Dordrecht : Kluwer Academic Publishers.

OKSALA, T. 1994. KAAD: Evolutionary and cognitive aspects, in *Knowledge-based computer-aided architectural design*. Amsterdam : Elsevier.

OXMAN, R. & Oxman, R. 1994. Case-based design: cognitive models for case libraries, in *Knowledge-based computer-aided architectural design*. Amsterdam : Elsevier.

POPOV, E.V. 1982. *Talking with computers in natural language*. Berlin : Springer-Verlag.

PUGH, S. 1996. *Creating innovative products using total design*. Reading, MA : Addison-Wesley.

RICHENS, P. 1994. Does knowledge really help? CAD research at the Martin Centre, in *Knowledge-based computer-aided architectural design*. Amsterdam : Elsevier.

RIESBECK, C.K. 1996. What next? The future of case-Based Reasoning in Post-Modern AI, in *Case-Based Reasoning* edited by D.B. Leake. Menlo Park, California : AAAI Press.

RISSLAND, E.L., Basu, C., Daniels, J.M., Rubenstein, Z.B. & Skalak, D.B. 1991. A Blackboard-based Architecture for CBR: An Initial Report, in *Proceedings of Case-Based Reasoning Workshop Sponsered by Defence Research Projects Agency Information Science and Technology Office*. San Mateo, California : Morgan Kaufman Publishers.

ROSS, P.J. 1988. *Taguchi techniques for quality engineering*. New York : McGraw-Hill.

RUMBAUGH, J., Blaha, M. Premerlani, W., Eddy, F. & Lorensen, W. 1991. *Object-oriented modelling and design*. Englewood Cliffs, New Jersey : Prentice-hall.

SAATY, T.L. 1980. *The analytic hierarchy process*. New York : McGraw-Hill.

SENGUPTA, A., Wilson, D.C. & Leake D.B. 1999. *On constructing the right sort of CBR implementation.* Computer Science Department, Indiana University.

SHEA, K. & Cagan, J. 1998. Generating structural essays from languages of discrete structures, in *Artificial intelligence in design*. Dordrecht : Kluwer Academic Publishers.

SIMOFF, S.J. & Maher, M.L. 1998. Designing with the activity/ space ontology, in *Artificial intelligence in design.* Dordrecht : Kluwer Academic Publishers.

SIMINA, M. 1999. *Enterprise-directed reasoning: opportunism and deliberation in creative reasoning*. PhD thesis, Atlanta : Artificial Intelligence Group College of Computing, Georgia Institute of Technology.

SMITH, PG. & Reinertsen, D.G. 1998. Developing Products in Half the Time. New York : John Wiley & Sons.

SNIDER, J.G. & Osgood, C.E. 1999. Semantic differential technique, in *Kansei Engineering Workshop at the eleventh symposium on Quality Function Deployment*. Novi, Michigan : QFD Institute.

SPARRIUS, A. 1998. Specification Practices, in *Course documentation, Ad Sparrius System Engineering and Management (Pty) Ltd*. South Africa, June 1998, Armscor Training Centre.

TECHNOSOLVE, CSIR. 1998. World class products and services with Quality Function Deployment, in *Workshop Guide* presented collaboratively by TechnoSolve and Aerotek, CSIR.

TERNINKO, J. 1999. Socially responsible QFD, in *Transactions from the eleventh symposium on Quality Function Deployment*. Novi, Michigan : QFD Institute.

ULRICH, KT. & Eppinger, S.D. 1995. *Product design and development*. New York : McGraw-Hill.

WATSON, I. 1997. *Appying Case-Based Reasoning.: Techniques for enterprise systems*. San Francisco, California : Morgan Kaufmann Publishers.

WHEELWRIGHT, S.C. & Clark, K.B. 1992. *Revolutionizing product development*. New York : The Free Press.

WILKE, W., Smyth, B. & Cunningham, P. 1998. Using configuration techniques for adaptation, in *Lecture Notes in artificial intelligence: Case-based reasoning technology*. Berlin : Springer-Verlag.

WORLD WIDE WEB CONSORTIUM. 1998. Extensible Markup Language (XML) 1.0. W3C Recommendation 10 February 1998. *Internet:* http://www.w3.org/TR/1998/REC-xml-19980210.html . Accessed: 18 July 2000.

WORLD WIDE WEB CONSORTIUM. 2000. Scalable Vector Graphics (SVG) 1.0 Specification. W3C Working Draft 29 June 2000. *Internet:* http://www.w3.org/TR/2000/WD-SVG-20000629/index.html . Accessed: 18 July 2000.

WORLD WIDE WEB CONSORTIUM. 1999. XML Fragment Interchange. W3C Working Draft 30 June 1999. *Internet:* http://www.w3.org/1999/06/WD-xml-fragment-19990630.html . Accessed: 18 July 2000.

WORLD WIDE WEB CONSORTIUM. 2000. XML Query Data Model. W3C Working Draft 11 May 2000. *Internet:* http://www.w3.org/TR/2000/WD-query-datamodel-20000511 . Accessed: 18 July 2000.

YOSHIOKA, M., Oosaki, M. & Tomiyama, T. 1996. An application of quality function deployment to functional modelling in a knowledge intensive design environment, in *Knowledge intensive CAD Volume 1*. London : Chapman & Hall.

ZIEMKE, M.C. & Spann, M.S. 1993. Concurrent engineering's roots in the World War II era, in *Concurrent Engineering contemporary issues and modern design tools*. London : Chapman & Hall.

ZULTNER, R.E. 1999. Defining customer needs for brand new products, in *Transactions from the eleventh symposium on Quality Function Deployment*. Novi, Michigan : QFD Institute.

ZVEGINTSEV, V.A. 1976. *Predlozzheye I yevo otnoshenye k yaziku i rechi*. Moscow : Mosovskiy Universitet.

# Appendix A: Implicit linking in PREMIS

The method of implicit linking was used in the precedent system PREMIS as a means of connecting alphanumeric information in a relational database to the graphic objects in an indexed graphical object library. This was a very convenient means of connecting diverse sources of information together. Consider Code Fragment 6 below that contains the ASCII representation of a simple rectangular graphic entity in PREMIS. The `subject` label indicates that the subject under consideration is related to `SPACE`. The `object` label indicates that the space is called `FS28:B7:F0:E2`. This is essentially a hierarchy that indicates that the graphical object describes the shape of a room `E2` that occurs on floor `F0` in a building called `B7` and a facility identified as `FS28`. The `:` separates the different parts of the object name or facets. It is interesting to note that this type of hierarchy fits naturally into the modern XML hierarchical paradigm. At the time when this format was used the processing speed of computers were such that raw ASCII code would have been inefficient. The fragment below was compiled into an efficient binary format and indexed with a highly optimised hashing procedure. The efficiency of current computers makes the use of structured ASCII code such as HTML, XML and the code fragment below feasible. The use of ASCII coding huge advantages such as:

- Very easy to read and understand
- Non-proprietary neutral knowledge formats that can be interpreted by any compliant software applications
- Very long life of data that can easily outlive the application that originally created it

```
. . .
subject
SPACE
object
FS28:B7:F0:E2
Hook
43.814300 –149.229000 0.000000
extents
1
4
40.965500 –149.532200 0.000000
44.323400 –152.048800 0.000000
46.663200 –148.926800 0.000000
43.305300 –146.410200 0.000000
drawing
E2
Scale
1.000000
rotation
0.000000 1.000000
world
0.000000 0.000000 0.000000
. . .
```

Code Fragment 6: Structure of a typical PREMIS graphical record

To connect a relational database record to the graphical record it is only necessary to create a database table record with four not null keys in a database such as Oracle or SQLServer. Relational database technology ensures that the combination of the key fields will always be unique. Two further constraints were placed on the database records:

The key fields must only contain uppercase characters
All key fields must have values (not null)

If during a query a user wants to display all graphical records related to the database a very simple SQL statement could be used such as:

```
SELECT
Space.SiteId||’:’||Space.BuildingId||’:’||Space.FloorId||’:’||Space.S
paceId FROM Space WHERE SiteId = ‘FS28’
```

By means of the concatenation of the key fields in abovementioned statement the graphical and alphanumeric records are logically related. This method is known as *implicit linking* because graphical and alphanumeric records are related by virtue of the similarity in the names. This offers the following important advantages:

- Data from diverse sources can easily be related together
- One alphanumeric relational database record can have multiple graphical representations ranging from outline to highly detailed
- Different operators (knowledge workers) can create the information knowing that it is logically related
- Information can conveniently be exported and imported from diverse distributed environments

The disadvantage of this method is that classification system must still be agreed on beforehand. Facilities managers have to decide what the codes should be and the graphical records must be structured in a similar way. This can nowadays be overcome by using a *Global Unique Identifier* (GUID) such as used in ActiveX controls. This provides the ultimate in globally unique codes. The only drawback of a GUID is that the code is non-mnemonic of nature making it difficult to know on face value what it relates to.

# Appendix B: PREMIS search criteria definition

The Following logic has been used for the search area definition in PREMIS.

Consider polygon $p_1 \ldots p_n$ and an arbitrary polygon *s.*

Define a function *T (s,i)* = {
*O* if *s* is outside $p_i$
*I* if *s* is inside $p_i$
*C* if *s* if s crosses $p_i$

Union

| | |
|---|---|
| Inside: | *i* 0 ⎤*n : T(s,i) = I* |
| Inside crossing: | *i* 0 ⎤*n : T(s,i)* 0 {*I,C*} |
| Crossing: | *i* 0 ⎤*n : T(s,i) = C* ϖ *i* 0 ⎤*n : T(s,i)* 0 {*O,C*} |

Intersection

| | |
|---|---|
| Inside: | *i* 0 ⎤*n : T(s,i) = I* |
| Inside crossing: | *i* 0 ⎤*n : T(s,i)* 0 {*I,C*} |
| Crossing: | *i* 0 ⎤*n : T(s,i)* 0 {*I,C*} ϖ *i* 0 ⎤*n : T(s,i) = C* |

Excluded intersection

| | |
|---|---|
| Inside: | *i* 0 ⎤*n : T(s,i) = I* ϖ *i* 0 ⎤*n : T(s,i) = O* |
| Inside crossing: | *i* 0 ⎤*n : T(s,i)* 0 {*I,C*} ϖ *i* 0 ⎤*n : T(s,i)* 0 {*O,C*} |
| Crossing: | *i* 0 ⎤*n : T(s,i) = C* ϖ ( *i* 0 ⎤*n : T(s,i)* 0 {*I,C*} ω *i* 0 ⎤*n : T(s,i)* 0 {*O,C*}) |

# Appendix C: Interface an ActiveX control to an Excel spreadsheet

```
Private Sub ArgosAB_GotFocus()
    Worksheets("Sheet1").Range("A1:A10").Value = ArgosAB.GrossArea
    txtArgos.Text = ArgosAB.GrossArea
End Sub

Private Sub ArgosAB_LostFocus()
    Worksheets("Sheet1").Range("A1:A10").Value = ""
    txtArgos.Text = "RESET TO EMPTY"
End Sub
```

# Appendix D: XSL stylesheet to convert XML into VML for web page display

```
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:v="urn:schemas-microsoft-com:vml">

   <xsl:template match="MicroGDS">
      <xsl:variable name="LX">
         <xsl:for-each select="Layer/Extent">
            <xsl:sort select="number(@LX)" data-type="number" order="ascending"/>
            <xsl:if test="position()=1">
               <xsl:value-of select="number(@LX)"/>
            </xsl:if>
         </xsl:for-each>
      </xsl:variable>

      <xsl:variable name="LY">
         <xsl:for-each select="Layer/Extent">
            <xsl:sort select="number(@LY)" data-type="number" order="ascending"/>
            <xsl:if test="position()=1">
               <xsl:value-of select="number(@LY)"/>
            </xsl:if>
         </xsl:for-each>
      </xsl:variable>

      <xsl:variable name="HX">
         <xsl:for-each select="Layer/Extent">
            <xsl:sort select="number(@HX)" data-type="number" order="descending"/>
            <xsl:if test="position()=1">
               <xsl:value-of select="number(@HX)"/>
            </xsl:if>
         </xsl:for-each>
      </xsl:variable>

      <xsl:variable name="HY">
         <xsl:for-each select="Layer/Extent">
            <xsl:sort select="number(@HY)" data-type="number" order="descending"/>
            <xsl:if test="position()=1">
               <xsl:value-of select="number(@HY)"/>
            </xsl:if>
         </xsl:for-each>
      </xsl:variable>

      <xsl:variable name="SX">
         <xsl:value-of select="$HX - $LX"/>
      </xsl:variable>

      <xsl:variable name="SY">
         <xsl:value-of select="$HY - $LY"/>
      </xsl:variable>

      <xsl:variable name="WW">
         800
      </xsl:variable>

      <xsl:variable name="HH">
         <xsl:value-of select="$WW * $SY div $SX"/>
      </xsl:variable>

      <v:group
         style="position: absolute; margin-left: 10px; margin-top: 10px; width:
{$WW}px; height: {$HH}px;"
         coordsize="{$SX},{$SY}"
         coordorigin="{$LX},{$LY}"
      >

      <!--frame-->

      <v:polyline
         points="{$LX},{$LY},{$HX},{$LY},{$HX},{$HY},{$LX},{$HY},{$LX},{$LY}"
      />

      <!--layer-->
```

```
<xsl:for-each select="Window/Phase">
   <xsl:if test="not(@State[.='Invisible'])">

      <xsl:variable name="LinkNo">
         <xsl:value-of select="@Layer"/>
      </xsl:variable>

      <xsl:for-each select="../../Layer">

         <xsl:if test="@LinkNumber[.= $LinkNo]">

            <!--lines-->
            <xsl:for-each select="OCD/Object/LinePrimitive">

               <v:polyline filled="false">
                  <xsl:attribute name="points">
                     <xsl:for-each select="Polyline/Point">
                        <xsl:value-of select="number(@X)"/>,
                        <xsl:value-of select="$HY + $LY - number(@Y)"/>,
                     </xsl:for-each>
                  </xsl:attribute>
               </v:polyline>

            </xsl:for-each>
            <!--end line-->

            <!--text-->
            <v:shapetype id="TextPrim" coordsize="21600,21600"
               path="m0,-14400l21600,-14400e">

            <v:path textpathok="t" />
            <v:textpath on="t" fitshape="t" xscale="t"/>
            </v:shapetype>


            <xsl:for-each select="OCD/Object/TextPrimitive">

               <!--extent-->
               <xsl:variable name="CharLX">
                  <xsl:for-each select="Extent">
                     <xsl:value-of select="number(@LX)"/>
                  </xsl:for-each>
               </xsl:variable>
               <xsl:variable name="CharLY">
                  <xsl:for-each select="Extent">
                     <xsl:value-of select="$HY + $LY - number(@LY)"/>
                  </xsl:for-each>
               </xsl:variable>
               <xsl:variable name="CharHX">
                  <xsl:for-each select="Extent">
                     <xsl:value-of select="number(@HX)"/>
                  </xsl:for-each>
               </xsl:variable>
               <xsl:variable name="CharHY">
                  <xsl:for-each select="Extent">
                     <xsl:value-of select="$HY + $LY - number(@HY)"/>
                  </xsl:for-each>
               </xsl:variable>
               <xsl:variable name="CharEX">
                  <xsl:value-of select="$CharHX - $CharLX"/>
               </xsl:variable>
               <xsl:variable name="CharEY">
                  <xsl:value-of select="$CharLY - $CharHY"/>
               </xsl:variable>

               <!--Char String-->
               <xsl:variable name="CharStringExpanded">
                  <xsl:value-of select="ExpandedText"/>
               </xsl:variable>
               <xsl:variable name="CharString">
                  <xsl:choose>
                     <xsl:when test="$CharStringExpanded=''">
                        <xsl:value-of select="DefinitionText"/>
                     </xsl:when>
                     <xsl:otherwise>
                        <xsl:value-of select="$CharStringExpanded"/>
                     </xsl:otherwise>
                   </xsl:choose>
               </xsl:variable>

               <!--CharStyle-->
               <xsl:variable name="CharFontType">
                  <xsl:value-of select="@Charstyle"/>
               </xsl:variable>
               <xsl:variable name="CharFontFamily">
                  <xsl:choose>
                     <xsl:when test="$CharFontType=''">
                        Times New Roman
                     </xsl:when>
                     <xsl:when test="$CharFontType='DEFAULT'">
                        Times New Roman
                     </xsl:when>
                     <xsl:otherwise>
                        <xsl:for-each select="../../../../Styles/TTCharstyle">
                           <xsl:if test="@Name[.= $CharFontType]">
                              <xsl:value-of select="@FontName"/>
                           </xsl:if>
                        </xsl:for-each>
                     </xsl:otherwise>
                  </xsl:choose>
               </xsl:variable>

               <!--Italic-->
               <xsl:variable name="CharItalic">
                  <xsl:choose>
                     <xsl:when test="$CharFontType=''">
                        normal
                     </xsl:when>
                     <xsl:when test="$CharFontType='DEFAULT'">
                        normal
                     </xsl:when>
                     <xsl:otherwise>
                        <xsl:for-each select="../../../../Styles/TTCharstyle">
                           <xsl:if test="@Name[.=$CharFontType]">
                              <xsl:choose>
                                 <xsl:when test="@Italic='true'">
                                    italic
                                 </xsl:when>
                                 <xsl:otherwise>
                                    normal
                                 </xsl:otherwise>
                              </xsl:choose>
                           </xsl:if>
                        </xsl:for-each>
                     </xsl:otherwise>
                  </xsl:choose>
               </xsl:variable>

               <!--text align-->
               <xsl:variable name="CharJustify">
                  <xsl:value-of select="@Justification"/>
               </xsl:variable>
               <xsl:variable name="CharAlign">
                  <xsl:choose>
                     <xsl:when test="$CharJustify='BL'">
                        left
                     </xsl:when>
                     <xsl:when test="$CharJustify='CL'">
                        left
                     </xsl:when>
                     <xsl:when test="$CharJustify='TL'">
                        left
                     </xsl:when>
                     <xsl:when test="$CharJustify='BC'">
                        center
                     </xsl:when>
                     <xsl:when test="$CharJustify='CC'">
                        center
                     </xsl:when>
                     <xsl:when test="$CharJustify='TC'">
                        center
                     </xsl:when>
                     <xsl:when test="$CharJustify='BR'">
```

```
                              right
                           </xsl:when>
                           <xsl:when test="$CharJustify='CR'">
                              right
                           </xsl:when>
                           <xsl:when test="$CharJustify='TR'">
                              right
                           </xsl:when>
                        </xsl:choose>
                     </xsl:variable>

                     <v:shape type="#TextPrim"
                        style="position:absolute; top: {$CharLY} ; left: {$CharLX};
width: {$CharEX} ;height: {$CharEY};"
                        adj="0" fillcolor="black" strokeweight="1pt">

                     <v:fill method="linear sigma" focus="100%"/>
                     <v:textpath
                        style='font-family : {$CharFontFamily};
                               font-style  : {$CharItalic};
                               font-weight : normal;
                               v-text-align: {$CharAlign};
                               v-text-kern:t'
                         trim="t" fitpath="t" xscale="f" string="{$CharString}"/>
                     </v:shape>

                  </xsl:for-each>
                  <!--end text-->

                </xsl:if>
            </xsl:for-each>

         </xsl:if>
      </xsl:for-each>

      </v:group>
   </xsl:template>
</xsl:stylesheet>
```

# Appendix E: Visual Basic code to implement a minimal web browser

```
Option Explicit
Public StartingAddress As String
Dim mbDontNavigateNow As Boolean

Private Sub cboAddress_Click()
    If mbDontNavigateNow Then Exit Sub
    timTimer.Enabled = True
    brwWebBrowser.Navigate cboAddress.Text

End Sub

Private Sub cboAddress_KeyPress(KeyAscii As Integer)
    On Error Resume Next
    If KeyAscii = vbKeyReturn Then
        cboAddress_Click
    End If
End Sub

Private Sub cmdBack_Click()
    timTimer.Enabled = True
    brwWebBrowser.GoBack
End Sub

Private Sub cmdForward_Click()
    timTimer.Enabled = True
    brwWebBrowser.GoForward

End Sub

Private Sub cmdHome_Click()
    timTimer.Enabled = True
    brwWebBrowser.Navigate StartingAddress
End Sub

Private Sub cmdRefresh_Click()
    timTimer.Enabled = True
    brwWebBrowser.Refresh
End Sub

Private Sub cmdSearch_Click()
    timTimer.Enabled = True
    brwWebBrowser.GoSearch
End Sub

Private Sub cmdStop_Click()
    timTimer.Enabled = False
    brwWebBrowser.Stop
    Me.Caption = brwWebBrowser.LocationName
End Sub

Private Sub Form_Load()
    On Error Resume Next
    Me.Show
    Form_Resize

    StartingAddress = "http://conradie/welcome.htm"
    If Len(StartingAddress) > 0 Then
        cboAddress.Text = StartingAddress
        cboAddress.AddItem cboAddress.Text
        timTimer.Enabled = True
        brwWebBrowser.Navigate StartingAddress
    End If

End Sub

Public Sub brwWebBrowser_NavigateComplete(ByVal URL As String)

    Dim i As Integer
    Dim bFound As Boolean
    Me.Caption = brwWebBrowser.LocationName
    For i = 0 To cboAddress.ListCount - 1
        If cboAddress.List(i) = brwWebBrowser.LocationURL Then
            bFound = True
            Exit For
        End If
    Next i
    mbDontNavigateNow = True
    If bFound Then
        cboAddress.RemoveItem i
    End If
    cboAddress.AddItem brwWebBrowser.LocationURL, 0
    cboAddress.ListIndex = 0
    mbDontNavigateNow = False

End Sub

Private Sub Form_Resize()
    cboAddress.Width = Me.ScaleWidth - 100
    brwWebBrowser.Width = Me.ScaleWidth - 100
    brwWebBrowser.Height = Me.ScaleHeight - 200
End Sub

Private Sub timTimer_Timer()
    If brwWebBrowser.Busy = False Then
        timTimer.Enabled = False
            Me.Caption = brwWebBrowser.LocationName
        Else
            Me.Caption = "Working..."
        End If
End Sub
```

# Appendix F: Visual Basic code to implement ARGOS intelligent component

```
'Default Property Values:

Const m_def_BackColor = 0
Const m_def_ForeColor = 0
Const m_def_Enabled = 0
Const m_def_BackStyle = 0
Const m_def_BorderStyle = 0
Const m_def_AA_xdim = 1000
Const m_def_AA_ydim = 1000
Const m_def_AA_zdim = 1000
Const m_def_AA_scale = 1
Const m_def_AA_unit = "mm"
Const m_def_AE_construction_area = 1000
Const m_def_AE_cost = 1
Const m_def_AE_durability = 1
Const m_def_AE_energy_use = 1
Const m_def_AE_grossarea = 1
Const m_def_AE_nettarea = 1
Const m_def_AE_rentable_area = 1
Const m_def_AE_shape = 6
Const m_def_AE_volume = 1
Const m_def_AF_function = ""
Const m_def_AT_hearing = ""
Const m_def_AT_internal_sensitivity = ""
Const m_def_AT_recognition = ""
Const m_def_AT_sight = ""
Const m_def_AT_smell = ""
Const m_def_AT_taste = ""
Const m_def_AE_wall_space_ratio = 0.9

'Property Variables:

Dim m_BackColor As Long
Dim m_ForeColor As Long
Dim m_Enabled As Boolean
Dim m_Font As Font
Dim m_BackStyle As Integer
Dim m_BorderStyle As Integer
Dim m_AA_xdim As Double
Dim m_AA_ydim As Double
Dim m_AA_zdim As Double
Dim m_AA_scale As Double
Dim m_AA_unit As String
Dim m_AE_construction_area As Double
Dim m_AE_cost As Currency
Dim m_AE_durability As Double
Dim m_AE_energy_use As Double
Dim m_AE_grossarea As Double
Dim m_AE_nettarea As Double
Dim m_AE_rentable_area As Double
Dim m_AE_shape As Double
Dim m_AE_volume As Double
Dim m_AF_function As String
Dim m_AT_hearing As String
Dim m_AT_internal_sensitivity As String
Dim m_AT_recognition As String
Dim m_AT_sight As String
Dim m_AT_smell As String
Dim m_AT_taste As String
Dim m_AE_wall_space_ratio As Double

'Event Declarations:

Event Click()
Event DblClick()
Event KeyDown(KeyCode As Integer, Shift As Integer)
Event KeyPress(KeyAscii As Integer)
Event KeyUp(KeyCode As Integer, Shift As Integer)
Event MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
Event MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
Event MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=8,0,0,0
Public Property Get BackColor() As Long
    BackColor = m_BackColor
End Property

Public Property Let BackColor(ByVal New_BackColor As Long)
    m_BackColor = New_BackColor
    PropertyChanged "BackColor"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=8,0,0,0
Public Property Get ForeColor() As Long
    ForeColor = m_ForeColor
End Property

Public Property Let ForeColor(ByVal New_ForeColor As Long)
    m_ForeColor = New_ForeColor
    PropertyChanged "ForeColor"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=0,0,0,0
Public Property Get Enabled() As Boolean
    Enabled = m_Enabled
End Property

Public Property Let Enabled(ByVal New_Enabled As Boolean)
    m_Enabled = New_Enabled
    PropertyChanged "Enabled"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=6,0,0,0
Public Property Get Font() As Font
    Set Font = m_Font
End Property

Public Property Set Font(ByVal New_Font As Font)
    Set m_Font = New_Font
    PropertyChanged "Font"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=7,0,0,0
Public Property Get BackStyle() As Integer
    BackStyle = m_BackStyle
End Property

Public Property Let BackStyle(ByVal New_BackStyle As Integer)
    m_BackStyle = New_BackStyle
    PropertyChanged "BackStyle"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=7,0,0,0
Public Property Get BorderStyle() As Integer
    BorderStyle = m_BorderStyle
End Property

Public Property Let BorderStyle(ByVal New_BorderStyle As Integer)
    m_BorderStyle = New_BorderStyle
    PropertyChanged "BorderStyle"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=5
Public Sub Refresh()

End Sub

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,1000
```

```
Public Property Get AA_xdim() As Double
    AA_xdim = m_AA_xdim
End Property

Public Property Let AA_xdim(ByVal New_AA_xdim As Double)
    m_AA_xdim = New_AA_xdim
    PropertyChanged "AA_xdim"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,1000
Public Property Get AA_ydim() As Double
    AA_ydim = m_AA_ydim
End Property

Public Property Let AA_ydim(ByVal New_AA_ydim As Double)
    m_AA_ydim = New_AA_ydim
    PropertyChanged "AA_ydim"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,1000
Public Property Get AA_zdim() As Double
    AA_zdim = m_AA_zdim
End Property

Public Property Let AA_zdim(ByVal New_AA_zdim As Double)
    m_AA_zdim = New_AA_zdim
    PropertyChanged "AA_zdim"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,1
Public Property Get AA_scale() As Double
    AA_scale = m_AA_scale
End Property

Public Property Let AA_scale(ByVal New_AA_scale As Double)
    m_AA_scale = New_AA_scale
    PropertyChanged "AA_scale"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=13,0,0,mm
Public Property Get AA_unit() As String
    AA_unit = m_AA_unit
End Property

Public Property Let AA_unit(ByVal New_AA_unit As String)
    m_AA_unit = New_AA_unit
    PropertyChanged "AA_unit"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MappingInfo=lblDescription,lblDescription,-1,Caption
Public Property Get AA_name() As String
    AA_name = lblDescription.Caption
End Property

Public Property Let AA_name(ByVal New_AA_name As String)
    lblDescription.Caption() = New_AA_name
    PropertyChanged "AA_name"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,1000
Public Property Get AE_construction_area() As Double
    AE_construction_area = m_AE_construction_area
End Property

Public Property Let AE_construction_area(ByVal New_AE_construction_area As Double)
    m_AE_construction_area = New_AE_construction_area
    PropertyChanged "AE_construction_area"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=2,0,0,1
```

```
Public Property Get AE_cost() As Currency
    AE_cost = m_AE_cost
End Property

Public Property Let AE_cost(ByVal New_AE_cost As Currency)
    m_AE_cost = New_AE_cost
    PropertyChanged "AE_cost"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,1
Public Property Get AE_durability() As Double
    AE_durability = m_AE_durability
End Property

Public Property Let AE_durability(ByVal New_AE_durability As Double)
    m_AE_durability = New_AE_durability
    PropertyChanged "AE_durability"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,1
Public Property Get AE_energy_use() As Double
    AE_energy_use = m_AE_energy_use
End Property

Public Property Let AE_energy_use(ByVal New_AE_energy_use As Double)
    m_AE_energy_use = New_AE_energy_use
    PropertyChanged "AE_energy_use"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,1
Public Property Get AE_grossarea() As Double
    AE_grossarea = m_AE_grossarea
End Property

Public Property Let AE_grossarea(ByVal New_AE_grossarea As Double)
    m_AE_grossarea = New_AE_grossarea
    PropertyChanged "AE_grossarea"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,1
Public Property Get AE_nettarea() As Double
    AE_nettarea = m_AE_nettarea
End Property

Public Property Let AE_nettarea(ByVal New_AE_nettarea As Double)
    m_AE_nettarea = New_AE_nettarea
    PropertyChanged "AE_nettarea"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,1
Public Property Get AE_rentable_area() As Double
    AE_rentable_area = m_AE_rentable_area
End Property

Public Property Let AE_rentable_area(ByVal New_AE_rentable_area As Double)
    m_AE_rentable_area = New_AE_rentable_area
    PropertyChanged "AE_rentable_area"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,6
Public Property Get AE_shape() As Double
    AE_shape = m_AE_shape
End Property

Public Property Let AE_shape(ByVal New_AE_shape As Double)
    m_AE_shape = New_AE_shape
    PropertyChanged "AE_shape"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,1
```

```
Public Property Get AE_volume() As Double
    AE_volume = m_AE_volume
End Property

Public Property Let AE_volume(ByVal New_AE_volume As Double)
    m_AE_volume = New_AE_volume
    PropertyChanged "AE_volume"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=13,0,0,
Public Property Get AF_function() As String
    AF_function = m_AF_function
End Property

Public Property Let AF_function(ByVal New_AF_function As String)
    m_AF_function = New_AF_function
    PropertyChanged "AF_function"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=13,0,0,
Public Property Get AT_hearing() As String
    AT_hearing = m_AT_hearing
End Property

Public Property Let AT_hearing(ByVal New_AT_hearing As String)
    m_AT_hearing = New_AT_hearing
    PropertyChanged "AT_hearing"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=13,0,0,
Public Property Get AT_internal_sensitivity() As String
    AT_internal_sensitivity = m_AT_internal_sensitivity
End Property

Public Property Let AT_internal_sensitivity(ByVal New_AT_internal_sensitivity As
String)
    m_AT_internal_sensitivity = New_AT_internal_sensitivity
    PropertyChanged "AT_internal_sensitivity"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=13,0,0,
Public Property Get AT_recognition() As String
    AT_recognition = m_AT_recognition
End Property

Public Property Let AT_recognition(ByVal New_AT_recognition As String)
    m_AT_recognition = New_AT_recognition
    PropertyChanged "AT_recognition"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=13,0,0,
Public Property Get AT_sight() As String
    AT_sight = m_AT_sight
End Property

Public Property Let AT_sight(ByVal New_AT_sight As String)
    m_AT_sight = New_AT_sight
    PropertyChanged "AT_sight"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=13,0,0,
Public Property Get AT_smell() As String
    AT_smell = m_AT_smell
End Property

Public Property Let AT_smell(ByVal New_AT_smell As String)
    m_AT_smell = New_AT_smell
    PropertyChanged "AT_smell"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
```

```
'MemberInfo=13,0,0,
Public Property Get AT_taste() As String
    AT_taste = m_AT_taste
End Property

Public Property Let AT_taste(ByVal New_AT_taste As String)
    m_AT_taste = New_AT_taste
    PropertyChanged "AT_taste"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MappingInfo=txtDescription,txtDescription,-1,Text
Public Property Get AA_description() As String
    AA_description = txtDescription.Text
End Property

Public Property Let AA_description(ByVal New_AA_description As String)
    txtDescription.Text() = New_AA_description
    PropertyChanged "AA_description"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,0.9
Public Property Get AE_wall_space_ratio() As Double
    AE_wall_space_ratio = m_AE_wall_space_ratio
End Property

Public Property Let AE_wall_space_ratio(ByVal New_AE_wall_space_ratio As Double)
    m_AE_wall_space_ratio = New_AE_wall_space_ratio
    PropertyChanged "AE_wall_space_ratio"
End Property

Private Sub cmd2D_3D_Click()
' Toggle the 2D - 3D mode command button
    Dim Isometric_y As Double
    Dim Isometric_x As Double

    Isometric_y = AA_ydim * 0.5
    Isometric_x = AA_ydim * 0.866

    If cmd2D_3D.Caption = "2" Then
        cmd2D_3D.Caption = "3"

        yzSlide.Value = AA_zdim

        UserControl.Height = AA_zdim + Isometric_y
        UserControl.Width = AA_xdim + Isometric_x
        UserControl.Height = AA_zdim + Isometric_y
        UserControl.Width = AA_xdim + Isometric_x

        cmdEnlarge.Top = Isometric_y + 25.1
        lblDescription.Top = Isometric_y + 25.1
        cmdReduce.Top = Isometric_y + 25.1
        cmd2D_3D.Top = Isometric_y + 361.446
        txtDescription.Top = Isometric_y + 361.446
        txtDescription.Height = AA_zdim - 680.72
        yzSlide.Top = Isometric_y + 25.1
        yzSlide.Height = AA_zdim - 149.498

        lblFrom.Top = UserControl.Height - 325.301
        lblCurrent.Top = UserControl.Height - 325.301
        lblTo.Top = UserControl.Height - 325.301
        xSlide.Top = UserControl.Height - 149.598

        shpDesign.Top = Isometric_y
        shpDesign.Height = UserControl.Height - Isometric_y

        ' Make the isometric projection lines visible

        linLine30_1.Visible = True
        linLine30_1.X1 = 0#
        linLine30_1.Y1 = Isometric_y
        linLine30_1.X2 = Isometric_x
        linLine30_1.Y2 = 0#

        linLine30_2.Visible = True
        linLine30_2.X1 = AA_xdim
        linLine30_2.Y1 = Isometric_y
        linLine30_2.X2 = UserControl.Width
        linLine30_2.Y2 = 0#

        linLine0_1.Visible = True
        linLine0_1.X1 = Isometric_x
        linLine0_1.Y1 = 0#
        linLine0_1.X2 = UserControl.Width
        linLine0_1.Y2 = 0#

        linLine90_1.Visible = True
        linLine90_1.X1 = UserControl.Width
        linLine90_1.Y1 = 0#
        linLine90_1.X2 = UserControl.Width
        linLine90_1.Y2 = UserControl.Height - Isometric_y

        linLine30_3.Visible = True
        linLine30_3.X1 = AA_xdim
        linLine30_3.Y1 = UserControl.Height
        linLine30_3.X2 = UserControl.Width
        linLine30_3.Y2 = UserControl.Height - Isometric_y


    Else

        cmd2D_3D.Caption = "2"

        yzSlide.Value = AA_ydim

        UserControl.Height = AA_ydim
        UserControl.Width = AA_xdim
        UserControl.Width = AA_xdim

        cmdEnlarge.Top = 25.1
        lblDescription.Top = 25.1
        cmdReduce.Top = 25.1
        cmd2D_3D.Top = 361.446
        txtDescription.Top = 361.446
        yzSlide.Top = 25.1

        lblFrom.Top = UserControl.Height - 325.301
        lblCurrent.Top = UserControl.Height - 325.301
        lblTo.Top = UserControl.Height - 325.301
        xSlide.Top = UserControl.Height - 149.598
        shpDesign.Top = 0#
        shpDesign.Height = UserControl.Height
        txtDescription.Height = UserControl.Height - 680.72
        yzSlide.Height = UserControl.Height - 149.498
        xSlide.Top = UserControl.Height - 149.498

        ' Set the projection lines invisible

        linLine30_1.Visible = False
        linLine30_2.Visible = False
        linLine30_3.Visible = False
        linLine0_1.Visible = False
        linLine90_1.Visible = False

    End If
End Sub

'Initialize Properties for User Control
Private Sub UserControl_InitProperties()

    m_BackColor = m_def_BackColor
    m_ForeColor = m_def_ForeColor
    m_Enabled = m_def_Enabled
    Set m_Font = Ambient.Font
    m_BackStyle = m_def_BackStyle
    m_BorderStyle = m_def_BorderStyle
    m_AA_xdim = m_def_AA_xdim
    m_AA_ydim = m_def_AA_ydim
    m_AA_zdim = m_def_AA_zdim
    m_AA_scale = m_def_AA_scale
    m_AA_unit = m_def_AA_unit
    m_AE_construction_area = m_def_AE_construction_area
    m_AE_cost = m_def_AE_cost
    m_AE_durability = m_def_AE_durability
    m_AE_energy_use = m_def_AE_energy_use
    m_AE_grossarea = m_def_AE_grossarea
    m_AE_nettarea = m_def_AE_nettarea
    m_AE_rentable_area = m_def_AE_rentable_area
    m_AE_shape = m_def_AE_shape
    m_AE_volume = m_def_AE_volume
    m_AF_function = m_def_AF_function
    m_AT_hearing = m_def_AT_hearing
    m_AT_internal_sensitivity = m_def_AT_internal_sensitivity
    m_AT_recognition = m_def_AT_recognition
    m_AT_sight = m_def_AT_sight
    m_AT_smell = m_def_AT_smell
    m_AT_taste = m_def_AT_taste
    m_AE_wall_space_ratio = m_def_AE_wall_space_ratio

End Sub

'Load property values from storage
Private Sub UserControl_ReadProperties(PropBag As PropertyBag)

    m_BackColor = PropBag.ReadProperty("BackColor", m_def_BackColor)
    m_ForeColor = PropBag.ReadProperty("ForeColor", m_def_ForeColor)
    m_Enabled = PropBag.ReadProperty("Enabled", m_def_Enabled)
    Set m_Font = PropBag.ReadProperty("Font", Ambient.Font)
    m_BackStyle = PropBag.ReadProperty("BackStyle", m_def_BackStyle)
    m_BorderStyle = PropBag.ReadProperty("BorderStyle", m_def_BorderStyle)
    m_AA_xdim = PropBag.ReadProperty("AA_xdim", m_def_AA_xdim)
    m_AA_ydim = PropBag.ReadProperty("AA_ydim", m_def_AA_ydim)
    m_AA_zdim = PropBag.ReadProperty("AA_zdim", m_def_AA_zdim)
    m_AA_scale = PropBag.ReadProperty("AA_scale", m_def_AA_scale)
    m_AA_unit = PropBag.ReadProperty("AA_unit", m_def_AA_unit)
    lblDescription.Caption = PropBag.ReadProperty("AA_name", "A")
    m_AE_construction_area = PropBag.ReadProperty("AE_construction_area",
m_def_AE_construction_area)
    m_AE_cost = PropBag.ReadProperty("AE_cost", m_def_AE_cost)
    m_AE_durability = PropBag.ReadProperty("AE_durability", m_def_AE_durability)
    m_AE_energy_use = PropBag.ReadProperty("AE_energy_use", m_def_AE_energy_use)
    m_AE_grossarea = PropBag.ReadProperty("AE_grossarea", m_def_AE_grossarea)
    m_AE_nettarea = PropBag.ReadProperty("AE_nettarea", m_def_AE_nettarea)
    m_AE_rentable_area = PropBag.ReadProperty("AE_rentable_area",
m_def_AE_rentable_area)
    m_AE_shape = PropBag.ReadProperty("AE_shape", m_def_AE_shape)
    m_AE_volume = PropBag.ReadProperty("AE_volume", m_def_AE_volume)
    m_AF_function = PropBag.ReadProperty("AF_function", m_def_AF_function)
    m_AT_hearing = PropBag.ReadProperty("AT_hearing", m_def_AT_hearing)
    m_AT_internal_sensitivity = PropBag.ReadProperty("AT_internal_sensitivity",
m_def_AT_internal_sensitivity)
    m_AT_recognition = PropBag.ReadProperty("AT_recognition", m_def_AT_recognition)
    m_AT_sight = PropBag.ReadProperty("AT_sight", m_def_AT_sight)
    m_AT_smell = PropBag.ReadProperty("AT_smell", m_def_AT_smell)
    m_AT_taste = PropBag.ReadProperty("AT_taste", m_def_AT_taste)
    txtDescription.Text = PropBag.ReadProperty("AA_description", "")
    m_AE_wall_space_ratio = PropBag.ReadProperty("AE_wall_space_ratio",
m_def_AE_wall_space_ratio)

End Sub

'Write property values to storage
Private Sub UserControl_WriteProperties(PropBag As PropertyBag)

    Call PropBag.WriteProperty("BackColor", m_BackColor, m_def_BackColor)
    Call PropBag.WriteProperty("ForeColor", m_ForeColor, m_def_ForeColor)
    Call PropBag.WriteProperty("Enabled", m_Enabled, m_def_Enabled)
    Call PropBag.WriteProperty("Font", m_Font, Ambient.Font)
    Call PropBag.WriteProperty("BackStyle", m_BackStyle, m_def_BackStyle)
    Call PropBag.WriteProperty("BorderStyle", m_BorderStyle, m_def_BorderStyle)
    Call PropBag.WriteProperty("AA_xdim", m_AA_xdim, m_def_AA_xdim)
    Call PropBag.WriteProperty("AA_ydim", m_AA_ydim, m_def_AA_ydim)
    Call PropBag.WriteProperty("AA_zdim", m_AA_zdim, m_def_AA_zdim)
    Call PropBag.WriteProperty("AA_scale", m_AA_scale, m_def_AA_scale)
    Call PropBag.WriteProperty("AA_unit", m_AA_unit, m_def_AA_unit)
    Call PropBag.WriteProperty("AA_name", lblDescription.Caption, "A")
    Call PropBag.WriteProperty("AE_construction_area", m_AE_construction_area,
m_def_AE_construction_area)
    Call PropBag.WriteProperty("AE_cost", m_AE_cost, m_def_AE_cost)
    Call PropBag.WriteProperty("AE_durability", m_AE_durability, m_def_AE_durability)
    Call PropBag.WriteProperty("AE_energy_use", m_AE_energy_use, m_def_AE_energy_use)
    Call PropBag.WriteProperty("AE_grossarea", m_AE_grossarea, m_def_AE_grossarea)
    Call PropBag.WriteProperty("AE_nettarea", m_AE_nettarea, m_def_AE_nettarea)
    Call PropBag.WriteProperty("AE_rentable_area", m_AE_rentable_area,
m_def_AE_rentable_area)
    Call PropBag.WriteProperty("AE_shape", m_AE_shape, m_def_AE_shape)
    Call PropBag.WriteProperty("AE_volume", m_AE_volume, m_def_AE_volume)
    Call PropBag.WriteProperty("AF_function", m_AF_function, m_def_AF_function)
    Call PropBag.WriteProperty("AT_hearing", m_AT_hearing, m_def_AT_hearing)
    Call PropBag.WriteProperty("AT_internal_sensitivity", m_AT_internal_sensitivity,
m_def_AT_internal_sensitivity)
    Call PropBag.WriteProperty("AT_recognition", m_AT_recognition,
m_def_AT_recognition)
    Call PropBag.WriteProperty("AT_sight", m_AT_sight, m_def_AT_sight)
    Call PropBag.WriteProperty("AT_smell", m_AT_smell, m_def_AT_smell)
    Call PropBag.WriteProperty("AT_taste", m_AT_taste, m_def_AT_taste)
    Call PropBag.WriteProperty("AA_description", txtDescription.Text, "")
    Call PropBag.WriteProperty("AE_wall_space_ratio", m_AE_wall_space_ratio,
m_def_AE_wall_space_ratio)

End Sub

Private Sub xSlide_Change()

    Dim Control_x As Double        ' Actual current total control width
    Dim Isometric_y As Double
    Dim Isometric_x As Double

    Isometric_y = AA - Ydim * 0.5
    Isometric_x = AA_ydim * 0.866

    ' The control is in 2D mode
    If (cmd2D_3D.Caption = "2") Then

        Control_x = xSlide.Value
        UserControl.Width = Control_x
        shpDesign.Width = Control_x
        lblDescription.Width = Control_x - 680.72
        cmdReduce.Left = Control_x - 427.71
        lblTo.Left = Control_x - 427.71
        lblCurrent.Left = (Control_x / 2#) - 190.771
        txtDescription.Width = Control_x - 445.783
        yzSlide.Left = Control_x - 149.498
        xSlide.Width = Control_x - 149.498

    'The control is in 3D mode
    Else

        Control_x = xSlide.Value
        UserControl.Width = Control_x + Isometric_x
        shpDesign.Width = Control_x

        lblDescription.Width = Control_x - 680.72
        cmdReduce.Left = Control_x - 427.71
        lblTo.Left = Control_x - 427.71
        lblCurrent.Left = (Control_x / 2#) - 190.771
        txtDescription.Width = Control_x - 445.783
        yzSlide.Left = Control_x - 149.498
        xSlide.Width = Control_x - 149.498

        linLine0_1.X2 = UserControl.Width
        linLine30_2.X1 = Control_x
        linLine30_2.X2 = UserControl.Width
        linLine90_1.X1 = UserControl.Width
        linLine90_1.X2 = UserControl.Width
        linLine30_3.X1 = Control_x
        linLine30_3.X2 = UserControl.Width

    End If

    AA_xdim = xSlide.Value

End Sub

Private Sub yzSlide_Change()

    Dim Control_y As Double         ' Actual current total control width
    Dim Control_z As Double         ' Actual current total control height
    Dim Isometric_y As Double
    Dim Isometric_x As Double

    Isometric_y = AA_ydim * 0.5
    Isometric_x = AA_ydim * 0.866

    ' The control is in 2D mode
    If (cmd2D_3D.Caption = "2") Then

        Control_y = yzSlide.Value
        UserControl.Height = Control_y
        shpDesign.Height = Control_y

        txtDescription.Height = Control_y - 680.72
        lblFrom.Top = Control_y - 325.301
        lblCurrent.Top = Control_y - 325.301
        lblTo.Top = Control_y - 325.301
        yzSlide.Height = Control_y - 149.498
        xSlide.Top = Control_y - 149.498

        AA_ydim = yzSlide.Value

    ' The control is in 3D mode
    Else
        Control_z = yzSlide.Value
        UserControl.Height = Control_z + Isometric_y
        shpDesign.Height = Control_z

        txtDescription.Height = Control_z - 680.72
        lblFrom.Top = (Control_z + Isometric_y) - 325.301
        lblCurrent.Top = (Control_z + Isometric_y) - 325.301
        lblTo.Top = (Control_z + Isometric_y) - 325.301
        yzSlide.Height = Control_z - 149.498
        xSlide.Top = (Control_z + Isometric_y) - 149.498

        linLine90_1.Y2 = Control_z
        linLine30_3.Y1 = UserControl.Height
        linLine30_3.Y2 = Control_z

        AA_zdim = yzSlide.Value

    End If

End Sub
```