USING THE UNIFIED MODELING LANGUAGE (UML) TO REPRESENT ARTEFACTS IN THE ZACHMAN FRAMEWORK

Mini-dissertation by LYNETTE REGINA ELS (83564595)

Submitted in partial fulfilment of the requirements for the degree

MASTER OF INFORMATION TECHNOLOGY
in the
SCHOOL OF INFORMATION TECHNOLOGY
of the
FACULTY OF ENGINEERING AND INFORMATION TECHNOLOGY
UNIVERSITY OF PRETORIA

University of Pretoria

Supervisor: Mr P. Joubert **November 2005**

Acknowledgements

I would not have been able to complete my mini dissertation without the help, guidance and support of the following people:

- My Study Leader, Pieter Joubert, for discussing and arguing my research topic and for guiding me through the research process.
- My colleagues at work, SITA and friends for all the support, advice and suggestions which contributed towards finalising my research proposal.
- My mother and sister for the valuable support just a family can provide when they encourage you to succeed.

Table of Contents

List of Tables

Table of Figures

I keep six honest serving men
(They taught me all I Knew):
Their names are What and Why and When
And How and Where and Who.

RUDYARD KIPLING, 1902.

USING THE UNIFIED MODELING LANGUAGE (UML) TO REPRESENT ARTEFACTS IN THE ZACHMAN FRAMEWORK

Abstract.

An interpretive research approach will be used to describe and decompose UML diagrams into their respective building blocks. A top down approach will be used to determine views that are important to enterprises during the system development lifecycle. The importance of providing graphical representations to describe conceptual ideas will be stressed. A short history will be provided of the origins of UML as well as a description of the diagrams used. Since UML is a language and not a methodology a brief discussion regarding a methodology, the Rational Unified Process, will be covered.

The Zachman framework will be used to present a two-dimensional (Columns and Rows) view of an enterprise together with a summary of what could be represented in the framework. The UML building blocks will be mapped within the Zachman framework together with possible reasons for the mapping.

The paper will conclude by combining several views by different authors to represent artefacts within the Zachman framework and to show the strengths and weaknesses of the current UML version 1.5 and what organisations should be aware of when considering implementing UML.

Keywords: Enterprise Architecture. Zachman. Unified Modeling Language. UML. Rational Unified Process. RUP, artefact, primitive, composite.

1. Chapter 1. Introduction.

1.1. Background.

Having a traditional programming and systems analysis background it was interesting to note that although many ICT projects have failed, many successful ICT systems have also been implemented using traditional dataflow diagrams and functional decomposition. Evidence of this is the legacy systems, rich in functionality that organisations still use today. These systems were implemented in the past 15 to 20 years and some are even older than that (Schach 2005:490). In the mid 1970s to 1980s a more structured approach using structured techniques was used for developing systems. Although successful in some cases the structured techniques lacked the capacity for coping with large or enterprise systems. The focus was either on functions or data, but they were not addressed simultaneously (Schach, 2005:18,19). Other systems development techniques such as the movement towards Object Orientation have since evolved towards the promise to design systems more efficiently and effectively by combining the functional and data focus at the same time with equal importance (Schach 2005:19). These techniques started at a detailed level to help transform conceptual ideas into system concepts that could be implemented as ICT systems that people and organisations could use to improve productivity.

1.2. Personal Experiences.

My background in IT started when the structured development techniques gained popularity with the use of Dataflow and Entity Relationship Diagrams. I started working with structured methods in a mainframe environment.

In retrospect, when analysing all the unsuccessful ICT projects I realised that one of the factors contributing towards the failures was the fact that these systems did not always add value towards the business and that some functionality was never used by either the users or the customers. Other problems were also identified as organisations or enterprises became bigger with more specialised functions. As my experience increased it became important to me that business knowledge should be shared extensively within enterprises in order to manage the business holistically from a top-down perspective. I realised how valuable the use of diagrams was to describe certain functionality available to the user or when certain adaptations to the current systems were required to accommodate changes in user requirements. Process Flow Diagrams, Functional Decompositions together with Data Flow Diagrams as well Data Models were used in an attempt to create a holistic representation of the requirements and how the system could address those requirements. The techniques were used from the implementation stage up to the maintenance activities of the implemented systems.

I realised one of the most important areas that I needed to focus on, was to spend more time on identifying the correct user requirements and that IT systems had to use more functionality and data items between IT systems. These aspects were also identified by various authors together with possible solutions on how to address the issues (Firesmith 2005:27-43; Spewak 1992:38). Firesmith highlighted the importance of incremental and iterative development cycles whilst Spewak promoted the notion of enterprise architecture planning.

1.3. Problem Statement.

If things that are important to the business are understood in the correct context, and if system requirements are successfully translated from those business ideas, it would enhance the successful implementation and utilisation of IT systems. A mechanism must be identified to make it possible. It is important that all the elements must work together to support the sustainability of IT systems. IT systems must support the business. To make it possible business requirements must be identified and captured as soon as possible to be available to the system developers. It must also be revisited and reviewed on a continuous basis by various stakeholders in the business -, system development -, IT infrastructure - and communications fields to ensure that the business requirements were correctly interpreted. Techniques must be investigated to bring IT and the business closer together to enhance the understanding and interpretation of elements that could be supported by IT systems.

Could concepts at a strategic and business level as well as at a more detailed systems development and design level be represented and captured using the UML? Formal system development techniques, such as the UML, used at a business level could provide system developers with a better understanding of the most important business elements that could be further enhanced into detail system requirements and models. By using this approach important business concepts could be captured and retained at a level that could be further transformed and enhanced to implement successful ICT projects.

1.4. Research Approach.

A qualitative research approach as opposed to quantitative research approach was adopted. An exact measurement in the research was not possible since cognitive reasoning is an important consideration in creating models. Two models will not be created exactly in the same way. During my research objective logical deduction was done to reach certain conclusions and to minimize any possible subjectivity. At the same time other similar models obtained during the research were verified and questioned by conclusions reached at the end of this research. Viewpoints about similar research-philosophies are further described by Martin Olivier in his book about Information Technology Research (Olivier 2004:109,110).

The aim of the research was to first understand the Zachman framework and the diagrams used in the UML and to relate the two by a process of logical reasoning. By using the logical technique the approach seemed to suggest that an interpretive type of research would be appropriate. The qualitative method was used will be a semiotics method whereby the concepts of UML diagrams were mapped in the Zachman framework (Olivier 2004:109-112,115).

The study attempted to use the UML and to organise all the diagrams of the language within the Zachman framework to show the applicability of using UML to develop IT systems by all the various stakeholders thereby ensuring the success implementation of IT systems (Zachman, 2005).

1.5. Overview of the Research Paper.

The context of the research paper has been discussed and the rest of the research paper will cover the following main topics:

- UML. UML will be described together with examples of how organisations use UML as well as some high-level examples of the diagrams. The Object Management Group (OMG) UML version 1.5 will be used as the main reference when discussing UML although references from other sources will also be included.

- Rational Unified Process (RUP). A summary of the RUP phases will be provided since it is significant when discussing how UML could be used by organisations.
- Enterprise Architecture. The discussion will start with why it is important for organisations to adopt architecture as a mechanism to implement IT systems. A framework, the Zachman Framework will be discussed as an example of a mechanism that organisations could use together with some benefits of using the Zachman framework.
- Mapping UML and RUP within the Zachman Framework. RUP and UML will be mapped within the Zachman framework together with reasons of the categorisation. A summary will be provided of where UML seemed to be used within the Zachman framework.
- Concluding Remarks. The paper will conclude by addressing the columns and rows not addressed by a UML diagram as well as concluding for what purpose UML could be used and what it was particularly suited for.

2. Chapter 2. UML.

2.1. Modeling Techniques.

Using modeling techniques to represent processes and things are an important consideration in Information Technology. One of the reasons is that a lot of the elements used in Information Technology are conceptual making it difficult for people to visualise the end result. By using models it can help to communicate difficult concepts to everyone part of the development process. It will be possible for business managers and technical software developers to understand the complexity of Information Technology by realising all the elements involved in developing successful ICT systems. Models can focus a work session on a specific view of the ICT system to obtain and communicate ideas. It would also be easy to identify risk and integration opportunities early on in the process. It is important that the techniques must support a graphical as well as a narrative component. The narrative component will usually be a text based technique. (Cernosek and Naiburg 2004:1-3).

Members of the development team will all have different backgrounds and experiences. This makes it important to use a well recognised standard that is well supported and maintained that could be understood and interpreted by everyone. UML has been identified as such a technique.

2.2. Organisations Using UML.

Enterprises are starting to use and to experience the benefits of using UML (Calio et al 2000:641). The reuse capability of UML is listed as one of the benefits of using UML (Griss 1998:8-12).

Business risks are also being reduced after using UML (Wang and Cone 2001:164-168). A range of different enterprises such as Command and Control Systems, hospital - as well as production systems are starting to use UML to model their enterprise components (Aagedal and Milosevic, 1998:88; Tanaka et al 2001:188; Bastos and Ruiz 2002:3786).

One of the reasons that organisations are starting to use UML could also be that UML is well supported and widely used. The Object Management Group (OMG) is the custodian of UML which is a non-proprietary technique. OMG, in collaboration with various partners including amongst others IBM, Oracle, Rational Software and Microsoft maintain UML. Continuous development is currently underway concerning UML.

The UML can also be expanded or extended by means of developing profiles such as the “Profile for modeling quality of service and fault tolerance and mechanisms” (OMG 2004). Other profiles available on the OMG website are the UML Profile for Enterprise Application Integration (EAI) as well as the “UML Profile for Schedulability, Performance and Time”.

The research paper is using specifications of UML version 1.5 but UML 2.0 is also currently available. All the current specifications of UML can be obtained from their website www.omg.org. (OMG 2003:56-57).

2.3. UML History.

UML originated from the Object Orientation approach to system development (Alhir 1998:4). Strengths identified during the use of an object orientation approach also apply to UML. One of the main strengths of an object orientation approach is the reuse of components as described by various authors (Schach 2005:21; Lethbridge and Laganiere 2005:68,69; Bennett et al 2002:211-213). Other benefits using the UML are helping organisations with integrating concepts (Evans et al, 2005:166,167; Cernosek and Naiburg, 2004).

UML started towards the end of 1994, beginning of 1995 when Grady Booch, JIM Rumbaugh and Ivar Jacobson teamed up to start developing the Unified Method. Disciplines of the Object Modeling Technique and Object-Oriented Software Engineering were merged to form UML. Their vision was to develop a scalable modeling language that would incorporate conceptual as well as detailed technical elements. All the stakeholders that are part of the development process must be able to use UML. The stakeholders would usually include business people, analysts as well as software developers. (OMG 2003:55-56).

2.4. What is UML?

UML is a language using specific notations that is classified into a set of diagrams. These diagrams help with the process of visualising, presenting and documenting user requirements in a graphical format during the development of systems. The diagrams will contain amongst other important structural and design elements of software systems (OMG 2003:45; OMG 2005).

UML is identified as the de-facto standard by various sources (Seidewitz, 2003:28;Bjӧrkander and Kobryn, 2003). The use of UML is also criticized, but not discarded in total, with relation to the use of Use Cases (Feldman et al, 2003:193). A business Use Case was presented in the article to illustrate the interaction between the system and other role-players or stakeholders (Feldman et al, 2003:196).

The view of Bjӧorkander and Kobryn is that UML was intended to be used for “general-purpose modeling” but the focus shifted to address business process modeling which is considered a specialized field (Bjӧorkander and Kobryn, 2003:57). A series of upgrades and additions from UML1.x versions to UML 2.0 seemed to promise an improvement in presenting systems (Bjӧorkander and Kobryn, 2003:61). An important condition of successful implementations of UML 2.0 would however be the learning and understanding of the syntax and notations. UML 2.0 also seems easier to use. Two main views, that of “structure and behavior”, form the basis together with seven other views (“Classes, Component, Use Case, Collaboration, State Machine, Activity and Sequence”) is part of UML 2.0 (Evans et al, 2005:166,167). Note: The focus of the rest of the paper is not on UML 2.0.

2.5. UML Representations.

During the process of decomposing the UML diagrams, UML notations described in OMG UML version 1.5 will be used (OMG 2003:406,407). It is realised that other sources (i.e. tools and techniques) can have different UML representations but the representations described in version 1.5 would be the starting point. Other representations used would be indicated.

As mentioned the main focus would be on the UML representations when describing the UML diagrams. All the UML diagrams have the following four kinds of graphic representations:

- 1-D symbols or *icons*. The representation will be classified as one of the smallest building blocks of UML diagrams that could stand alone. An example would be *actors*.

- *2-D symbols.* These types of symbols usually include other symbols and will be considered as complex building blocks. Examples will be *sub systems* and *nodes*.
- Lines. Lines or *paths* contain a symbol at the start and end of a line and will be always be used together with other UML building blocks. Lines usually indicate relationships between the UML building blocks.
- Labels. Labels or *strings* are expressed in text using a specific language such as English or other formal languages using English such as the Object Constraint Language. Statements expressed in a specific programming language could also be used. These types of representations will be classified as one of the smallest UML building blocks.

(OMG 2003:406,407).

The classification as specified in UML version 1.5, with the exception of the Use Case Diagram, would be used when describing the UML diagrams. The Use Case Diagram would be classified as a diagram presenting a type of behaviour. The diagram names used in UML version 1.5 would also be used (OMG 2003:45,46,402-403; OMG 2005). The following diagrams will be discussed:

- Behaviour Diagrams (also known as Dynamic models). These diagrams would describe how objects change and evolve at certain points during the system life-cycle. It will contain aspects such as the different interactions between objects and how the objects change over a certain period (OMG 2003:61) The following diagrams will be discussed:
  - Use Case Diagram.
  - Sequence Diagram.
  - Collaboration Diagram.
  - Activity Diagram.
  - State Chart Diagram.
- Structure Diagrams (also known as static models). It describes the structure of objects and would include attributes and the relationship between objects (OMG 2003:61). The following diagrams are discussed:
  - Class Diagram.
  - Object Diagram.

- Implementation Diagrams. These diagrams are also seen as “Structural diagrams” (OMG 2005) describing how specific modules are combined with the purpose to implement and deploy systems successfully (OMG 2003:54). The following diagrams are discussed:
    - Component Diagram.
    - Deployment Diagram.

2.6. Behaviour or Dynamic Diagrams.

2.6.1. Use Case Diagram.

The purpose of the Use Case Diagram is to describe how the system is used. Functions together with external requirements are specified (Alhir 1998:161). The focus of the use case is to describe certain behaviour between the actor and the use case. The use case does not describe any structural aspect of the system but the use case must indicate that a result has been achieved by the actors (OMG 2003:194,198). All the business processes are identified (Satzinger et al 2004:245) and show how the actor “interacts with the system” (Pressman 2005:169). A specific notation is used for the Use Case Diagrams consisting of stick figures for the actors, ellipses for the functions and lines for the relationships (OMG 2003:494).

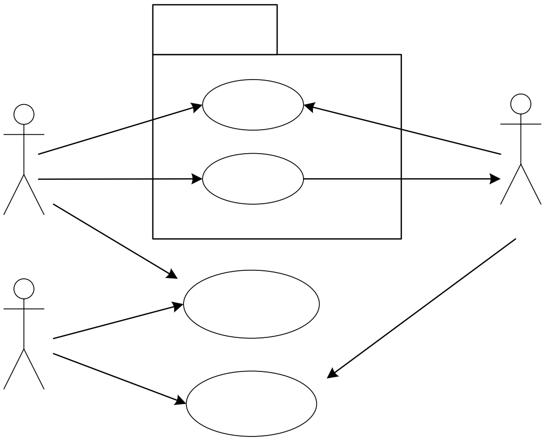


Figure 1: Use Case Diagram.

A Use Case Diagram contains a set of use cases (Schach 2005:503). Different scenarios are contained within a Use Case Diagram (Satzinger et al 2004:261; OMG 2003:717). Actors are any user of the system or role that they adopt when performing work (Satzinger et al 2004: 245). Actors provide input to or receive output from the function or process.

An important aspect of use cases is to describe the relationship or interaction that the function has with the environment in which it will be used. The actor is a part of the environment and the function is part of the software product that would be used by the actor (Schach 2005:276). The completeness of the use case could also be verified by a set of questions that must be answered (Pressman 2005:192).

Another viewpoint of the purpose of use cases is provided by Alistair Cockburn. His paper provides a theoretical background for use cases. When starting to specify use cases it is important to determine the goal of the function or what is to be accomplished by the function. He also explains that each actor has a set of responsibilities that must be accomplished thereby triggering or receiving information from certain functions. Actors are classified into primary and secondary actors. Primary actors need the system or sub-system to perform work and secondary actors only requires assistance. The purpose of use cases is to “gather user stories, or build requirements” (Cockburn, 2004).

An analyst usually specifies the Use Case Diagrams during user work or brain storming sessions when requirements are gathered. During this process the actors and functions would be identified (Satzinger et al 2004: 245; Schach 2005:389). As previously specified, a Use Case Diagram contains a set of use cases and different scenarios are contained within a use case. It could be concluded that a use case must be described by a set of scenarios.

Use Cases are specified at a summary or an overview level as well as at a detail level (Satzinger et al 2004: 245). At the summary level, a Use Case Diagram can be used grouped together in packages or sub systems. At a detail level, descriptions of the actors and functions are included. Pre- and post conditions that must exist when functions are performed, as well as any exceptions, should also be listed (Satzinger 2003:254; Whitten et el 2004:444).

Several templates are available to help with the narrative specification of use cases (Satzinger et al 2004:254; Whitten et al 446-453). Martin Fowler, when discussing use cases on his website, also identifies that the value of Use Cases lies in the narrative description and not the Use Case Diagram. (Fowler). Different levels of templates may be used from business use cases up to design use cases each with a different level of detail. The following headings as described by Whitten et al are present in these templates: (Whitten et al 2004:446-447)

- *Use Case Name and ID.*
- *Sources where information was obtained.*
- *Use Case Description.*
- *Interested stakeholders*.
- *Precondition.* What must exist before the use case can start?
- *Trigger*. What starts the use case initially?
- *Course of Events*. The steps of a typical scenario are described.
- *Alternative courses*. This can be an indication of a set of different responses that could occur.
- *Post Condition*. At the end of the use case, what is the desired state?
- *Assumptions*. A list of all the presumptions can be provided.
- *Business Rules*. Any rules or constraints can be provided.

Use cases also have the following benefits in the software development process as identified by Lethbridge and Laganiere:

- It helps with determining the boundaries of the system.
- It helps with the planning of the project to determine the complexity and size of the project.
- It helps to verify that all the requested requirements being developed are linked to a Use Case.
- It helps with determining a test strategy used during system and acceptance testing.
- It helps with the organising the user manuals.

However, although Use Cases may have value in determining user requirements, not all functional requirements such as performance related measures of the database or the archiving of data fields are covered by use cases. Developers must be aware of developing the system precisely according to the way the users are currently performing the functions. Better and innovative ways driven by the continuous expanding of ICT technology should be incorporated within Use Cases (Lethbridge and Laganiere 2005:139,140).

It must be stressed that Use Cases must not be used in isolation and must be combined with other techniques in the UML suite of techniques. Use Cases provide information to sequence and collaboration diagrams since use cases do not express any order or sequence of the functions. It does not show the different relationships between the actors as displayed by the collaboration diagram (Alhir 1998:161). Use Case Diagrams can also be complemented by a use case description, Activity Diagrams and Class Diagrams (Satzinger et al 2004:245,270).

2.6.2. Sequence Diagram.

Sequence diagrams are also known as System Sequence Diagrams (SSD) (Satzinger et al 2004:258). The purpose of sequence diagrams is to show how information that originates from actors is moved between objects. This movement is also called “interactions” (Alhir 1998:85) and is described by showing how messages are sent between objects and at what time messages could be initiated. Pressman describes sequence diagrams as events that are passed along to objects indicating an information flow between objects. The events are presented as a “function of time”. (Pressman 2005:251,252). In short, it shows how the various objects communicate with each other to accomplish specific tasks. The messages between objects must also be numbered to help with the sequencing of events (Alhir 1998:172). These messages can also be combined and be presented in sets of scenarios showing “real-time” situations (Alhir 1998:168,172; Satzinger et al 2004:258-261).

Sequence diagrams present a horizontal and vertical perspective of the objects. The horizontal view contains the objects and messages; and the vertical view shows how time influences the actions (Alhir 1998:168; Pressman 2005:252). Notation is available to indicate recursive calls as well as synchronous and asynchronous messages. In synchronous messages control is passed from the objects that initiate the messages to the object that must respond to it. The object waits for a response. This need may not be the case in asynchronous messages where actions may continue without waiting for a response. Messages may also be repeated many times (Alhir 1998:171-175).

A sequence diagram also indicates the steps of how an input that originates externally is changed or transformed to an output (Pressman 2005:252).

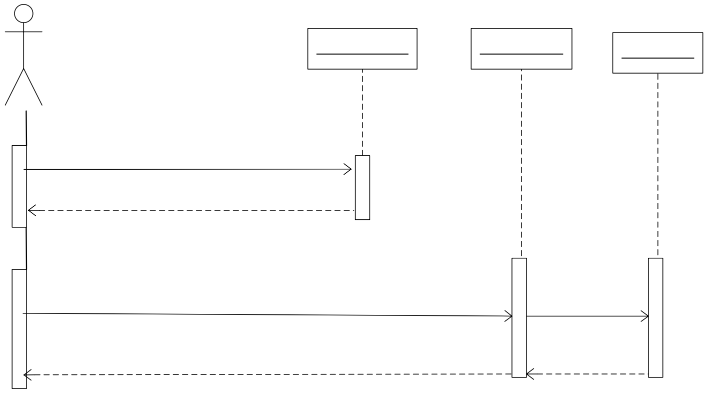


Figure 2: Sequence Diagram.

Sequence diagrams follow after use cases as they contain more detailed information than use cases with respect to how resources (actors and objects) use the system by means of messages to perform tasks (Alhir 1998:85, 168; Pressman 2005:251; Satzinger et al 2004:261). A user input is required since it shows how the users communicate with other actors. However a first draft of the sequence diagrams can be constructed by the analyst after specifying use cases. It is important to note that an iterative approach is adopted during the constructing of UML diagrams and after creating sequence diagrams, it may result in adding or combining certain use cases.

Sequence Diagrams, together with use cases, describe the "processing requirements for the system and give the foundation of system design." (Satzinger et al 2004: 265).

Sequence diagrams must be created in the context of collaboration diagrams that specify another viewpoint of the requirement (Alhir 1998:168).

Top Package::Customer

2.6.3. Collaboration Diagram.

The purpose of Collaboration diagrams is to understand who participates in performing tasks as well as to understand their relationships with each other (Alhir 1998:178). The types of associations or relationships (Schach 2005:197) used in the object orientated approach to system analysis and design will include multiplicity (Alhir 1998: 180). The Collaboration diagram shows how roles interact which each other by means of messages (Alhir 1998:94). The main focus of Collaboration

1. Verify Member

2. Display member data

3. Pla

Diagrams is on the objects as opposed to Sequence Diagrams that focus on the sequence of activities (Satzinger et al 2004:422). Collaboration diagrams also help to address coupling. (Satzinger et al 2004:420). Coupling is a design principle showing the interfaces or links between objects. Objects must be grouped together in order to minimize the interfaces between classes (Satzinger et al 2004:442).

It can be concluded that collaboration diagrams are related to sequence diagrams in respect of the notations used to specify objects and messages between the objects. They differ from sequence diagrams in that they do not show a horizontal or vertical perspective. Collaboration diagrams may contain a hierarchical view of how the different roles are organised to show interactions (Alhir 1998:179).

Collaboration diagrams may also be used at a design level where the description of messages may contain syntax specified in the Object Constraint Language (OCL) (Pressman 2005:340). Specifications describing the syntax can be found on the OMG website www.omg.org.

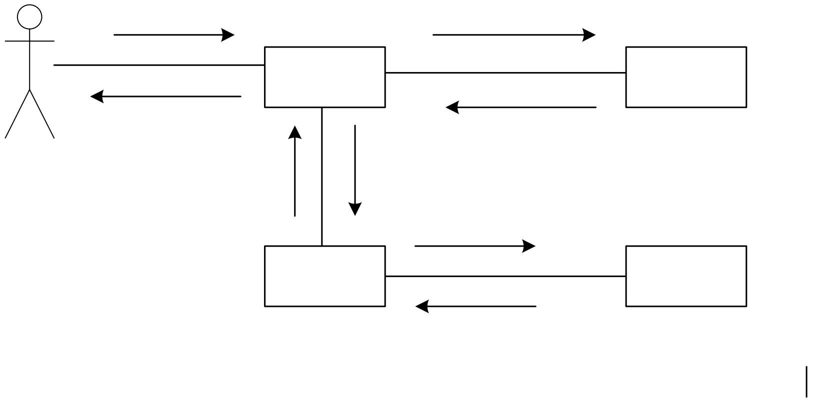


Figure 3: Collaboration Diagram.

2.6.4. Activity Diagram.

The purpose of activity diagrams is to understand the internal flow of information between activities of the system (Alhir 1998: 102). Inputs and outputs can easily be identified per activity and when obtaining information to create the diagram a process-oriented focus must be adopted (Satzinger et al 2004:261,262). The activity diagram is similar to a flowchart describing the steps that are necessary to perform activities (Pressman 2005:168). Swim lanes may be used to divide a set of activities with the same goal or responsibilities to accomplish the task. Actions and flows are described by the diagram together with a start and finish state of the activity (Alhir 1998:205-209; Pressman 2005:224-225). The notation includes synchronisation points illustrating that activities were performed at the same time, as well as decision points that could cause the flow to be split into different paths (Pressman 2005:168).

Activity diagrams may be constructed by the analyst and users in a work session.

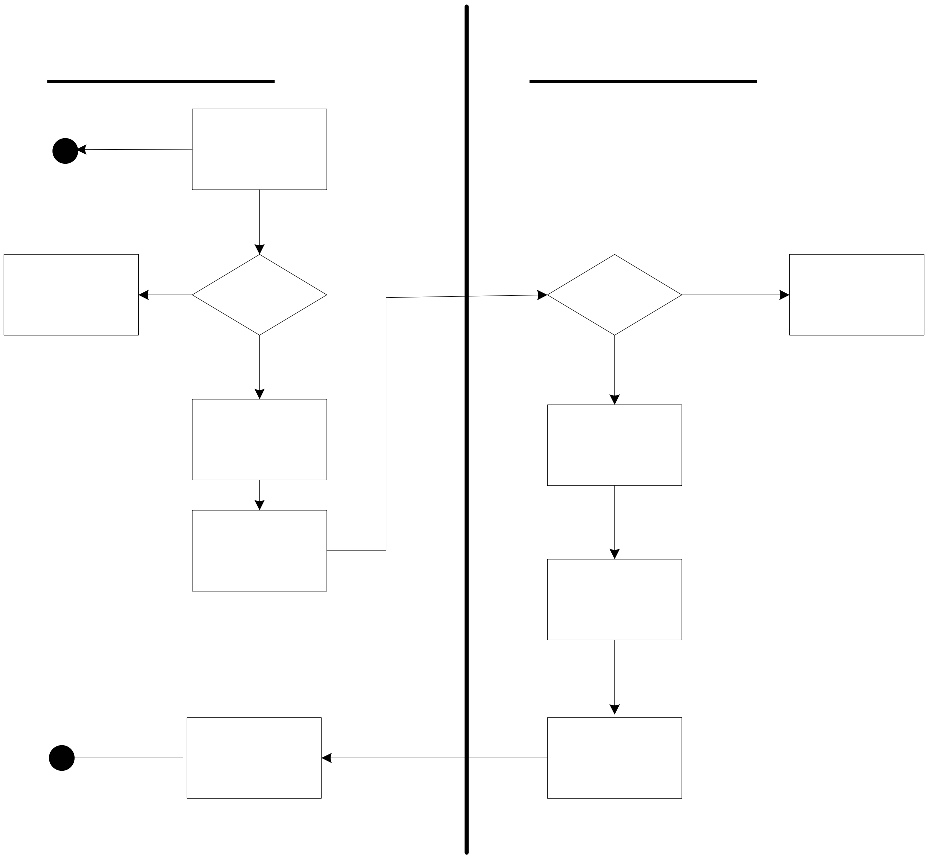


Figure 4: Activity Diagram.

Activity diagrams may also be used at a very low-level, i.e. just before the "source code" at a "component level design specification." (Pressman 2005:342).

2.6.5. State Chart Diagram.

State Chart Diagrams are also known as State Machines (Alhir 1998:187). The purpose of State Chart Diagrams are to show the different conditions that objects can be in during their life-cycle. The transitions between the states may be indicated by information flows (Alhir 1998 99-101). Pressman indicates that the transitions are "driven by events" Pressman 2005: 343). Descriptions of the transitions may include Boolean conditions, parameters indicating what triggers the action or any other test that may highlight the change in the object state (Alhir 1998:193).

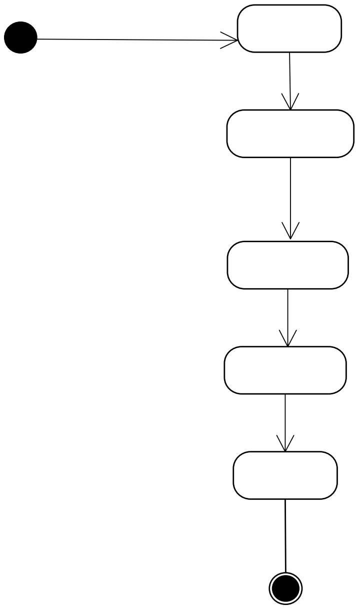


Figure 5: State Chart Diagram.

The analyst can construct the state chart diagram after the different objects are identified. During a work session the user may review the diagram.

The State Chart Diagram is related to the object diagram since it illustrates how objects can evolve during their life-cycle.

2.7. Structure of Static Diagrams.

2.7.1. Class Diagram.

Also known as a Class Model or an Object Model (Alhir 1998:81).The purpose of the class diagram is to show the structure of the system at a specific point in time, and not how the system changed between two periods (Alhir 1998:75). The class diagram can also be considered as the most important model. Every other model is validated against the class Diagram since objects cannot be used that are not included in the Class Diagram (Satzinger et al 2004:266).

The Class Diagram contains objects together with attributes and operations (Alhir 1998:140-143; Pressman 2005:169.170). Associations or relationships between objects are also described. The different associations that maybe used are borrowed from the Object Orientation paradigm and include aggregation, composition, multiplicity, generalisation and inheritance. The methods as advocated by the Object Orientation paradigm are described by operations in the class diagram (Alhir 1998:75-79).

Different levels of detail may be presented in a Class Diagram. Users may take part in constructing the first draft version of a class diagram. Details may be added during subsequent sessions or when the diagram is reviewed by the analyst. When adopting a system-engineering approach, classes are identified from the problem statement (Pressman 2005:168). This viewpoint highlights the fact that users should form part of identifying classes and objects.

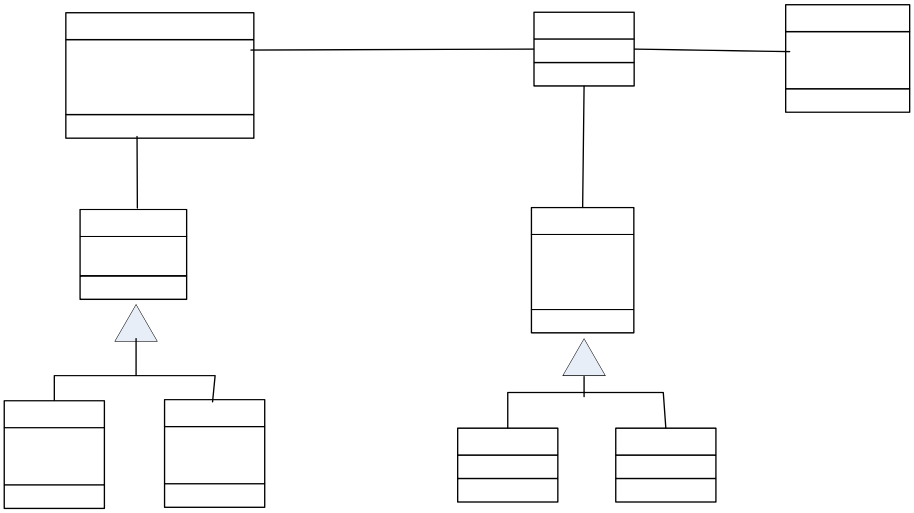


Figure 6: Class Diagram.

At a logical level class diagrams are also used to describe non-functional requirements of a system (Cysneiros and Prado Leite:2001). The non-functional requirements listed by Cysneiros include cost, reliability, security, portability, accuracy etc. The classification is also identified by Sommerville and it is extended to include usability, efficiency, and organisation requirements, legislative and ethical requirements (Sommerville:2004:122).

2.7.2. Object Diagram.

Object diagrams are similar to class diagrams but differ with respect to the scope of what is modelled. Object diagrams can be seen as a subset of class diagram. In Object diagrams a specific situation is described. Object and class diagrams are related since the one can be used to validate the other (Alhir 1998:82). The notation of object diagrams is also similar to class diagrams (Alhir 1998: 139-158). Usually instances of objects are modelled in object diagrams (Whitten 2004:441).

Product

1..

-productNr
-productDescription
-shelfnr
-productQuantity
+allocateShelf(productNr)()

Title

-titleDesciption
-catalogueNr

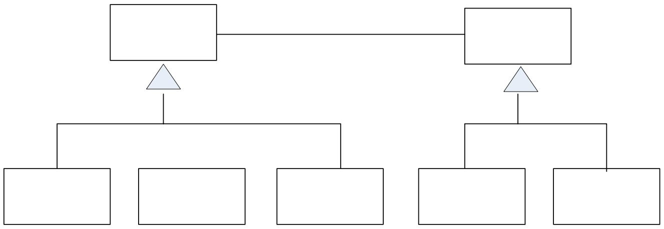


Figure 7: Object Diagram.

2.8. Implementation Diagrams.

2.8.1. Component Diagram.

The purpose of a component diagram is to describe the dependencies and interfaces between software components. Only types of components are indicated. A deployment diagram could be used where instances of components could be described (OMG 2003:569). Component Diagrams could also be used by programmers to specify how a software program is made up of different modules (Whitten et al 2004:442).

Product

The notation for software components are rectangular boxes with two smaller rectangular boxes at the side. Interfaces are shown by means of circles and the dependencies are dashed lines with arrows. Components do not contain attributes (Alhir 1998:104; OMG 2003:569-570). The physical units, source code, that make up a system are specified in a component diagram (Alhir 1998:211)

Book Title DVD Title Vide

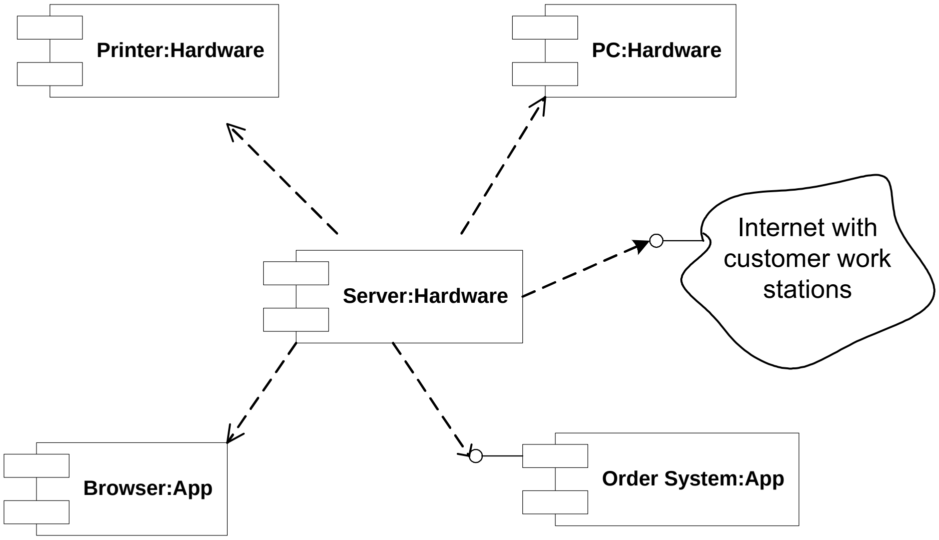


Figure 8: Component Diagram.

Due to the physical nature of the diagram, the analyst responsible to implement the system successfully would create the diagram with minimal or no user input at all. The user may be part of a review session after the analyst has interpreted other documents (i.e. technology standards etc) and diagrams (i.e. behaviour and static diagrams etc) to compile and relate the applicable components.

2.8.2. Deployment Diagram.

The purpose of the deployment diagram is to present a view of where physical components would be implemented in a specific environment. It shows the configuration of software components. Examples may include processing elements or "run-time software components" as described by Whitten et al) as well as software code. Deployment diagrams describe a view of the hardware architecture of a system (Whitten et al 2004:443; OMG 2003:571).

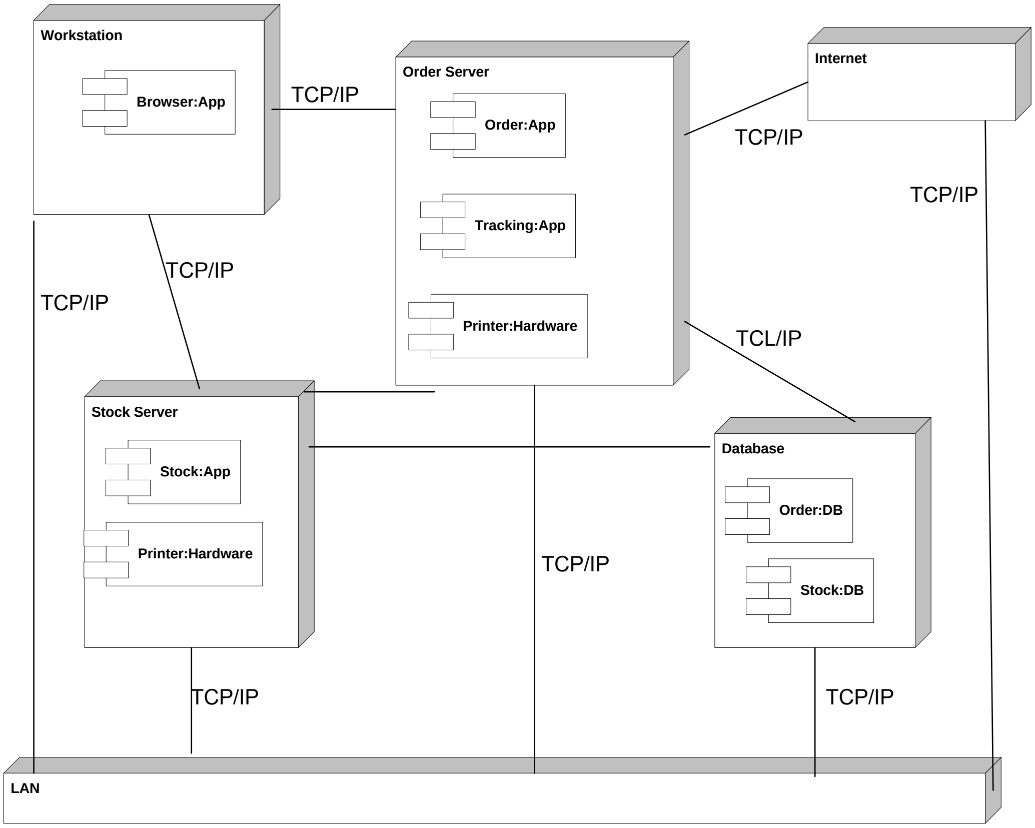


Figure 9: Deployment Diagram.

It is also suggested that deployment diagrams could be used to indicate how procedures and documents could be used by organisations. In this example the processing elements would be the employees and the software components the procedures and documents (OMG 2003:571).

The diagram contains a set of nodes or three dimensional boxes together with relationships. Nodes could be hardware resources or devices that have some kind of processing capability. Examples are workstations, printing devices, central servers etc. (OMG 2003:572,573; Alhir 1998:215,218; Whitten et al 2004:706-707).

Hardware elements and operating systems that would form part of the physical architecture of the system together with the location details could be modelled using a deployment diagram. The hardware elements could be Computer-Off –The-Shelf (COTS) products that must be integrated in the system (Pressman 2005:168, 345).

The designer or system engineer of the system would use this diagram with minimal user input.

3. Chapter 3. Rational Unified Process (RUP).

The RUP is a structured prescriptive methodology (Whitten et al 2004:100) or process model described by the software engineering discipline (Pressman 2005:77) or it is a set of activities required to control, organise, schedule, model, develop and implement projects (Pressman 2005:77,99; Sommerville 2004:64-65).

Generic process models used by projects are the *Waterfall Model*, the *Evolutionary Development* Model and *Component-based software engineering* (Sommerville 2004:64-65). The Rational Unified Process (RUP) is a modern process model that combines activities of all the three the generic process models. The RUP originated by the same persons responsible for developing the UML, i.e. Ivar Jacobson, Grady Booch and James Rumbaugh. Important views from the client or customer as well as the software architect are stressed. It also includes the concept of increments (to improve step-by-step) and iteration (to improve by successive new releases) as well as theories from the object-orientated paradigm is used. The RUP is now widely referred in the literature as simply the Unified Process (Pressman 2005:94-95; Schach 2004:23-24 48-49).

The RUP provides an indication of the sequence of activities together with the necessary documentation that can be performed using the UML. Another important benefit of using a prescribed methodology is that it will ensure that steps could be repeated during every project and by all the team members of project teams. Apart from helping the project manager with the scheduling of project steps it also helps with communicating the amount of work needed to implement the project (Rational Staff 2003).

The discussion of RUP is very significant whenever UML is mentioned, since UML is the language that supports a formal methodology or set of processes. The methodology that complements UML is the RUP (Alhir 1998:7; Schach 2005:498).

The main stages of the RUP process are *Inception, Elaboration, Construction and Transition* (Rational Staff 2003; Schach 2005:78-83; Pressman 2005:96-99; Sommerville 2004:82-85).

Inception. During this stage all the activities necessary to start a project are described. Activities such as confirming the main requirements, risks and constraints of the project are identified. The development process is specified in detail, as is the development infrastructure. Any plans and tools used to support the process are also specified. Any high-level or conceptual models are permitted to describe and provide context wherein the proposed system should operate in. Typical deliverables include a business case, risk and project plans, as well as an initial Use Case Diagram (IBM 2003; Pressman 2005:99).

Elaboration. The system development process starts during the Elaboration and includes activities such as creating design, class and implementation models. Important to note is that any models created during the inception phase are revisited and expanded to supply more detail. Architecture and technical risks are identified and the architecture environment wherein specific teams and tools will operate in is established. A preliminary design to support the requirements may also be created. Typical deliverables include revisited project and risk plans as well as use case and analysis models. Preliminary design models are also included (IBM 2003, Pressman 2005:99).

Construction. During this stage additional requirements are confirmed and the requirements are built. The main focus of this stage is the development of components that must be integrated and eventually be deployed and used by clients or customers. Acceptance criteria that were envisaged during the Inception stage must be revisited and updated. The main focus of this stage is to reach an achievable level of quality software components in the specified time and also to control the development costs. Typical deliverables include software components, test cases, user manuals, revisited design models and project and risk plans (IBM 2003, Pressman 2005:99).

Transition. The main focus of this stage is to ensure that the system is deployed for use by the clients or customers. Any user-feedback must be obtained and any changes must be controlled in order to achieve a workable product release. Activities such as the training as well as the “roll-out” to other departments are included (IBM 2003). Typical deliverables include test reports, user feedback and the developed software (IBM 2003, Pressman 2005:99)

4. Chapter 4. Enterprise Architecture.

The term architecture is used in various disciplines such as the construction of buildings, houses and bridges. It is also used and applied in the Information Technology field to describe the aspects necessary to build Information Systems. An enterprise architecture contains all the important building blocks together with models to show how all the building blocks are combined to form a specific structure. It can be considered as a tool that is used to describe different perspectives. (The Open Group, 2003:10). Part of the architecture building blocks are all the relationships necessary to describe how everything will fit together to form an Information System structure (Frankel et al 2003:1, The Open Group 2003:9). Architectures are made up from different components together with the different relationships between these components (Schach 2005:417). Various different architectures exist such as Business, Data, Application, Technology as well as Information System Architectures and enterprise architectures (Spewak 1992:1; The Open Group 2003:9, Zachman 1987). Software architectures as defined by Schach can be a combination of all the architectures already mentioned. It can also be extended to include non-functional requirements such as portability, reliability, maintainability as well as security (Schach 2005:417). The types of Software architecture are also described in the Software Engineering Institute website, http://www.sei.cmu.edu. The definition includes a structural and behaviour component as well as the relationships between the components (Software Engineering Institute, 2005).

Current organisations are shifting their focus to also include business systems together with ICT systems. These systems have their own architectures together with interactions between them (Aerts et al, 2004:781). When all these architectures are combined together it will result in a very complex enterprise architecture system that would have to be monitored and managed (Delen and Perakath 2003:257).

A technique identified by Delen et al to address the complexity issue is that of enterprise modeling (Delen and Perakath 2003:257-258). Models, together with modeling languages, are techniques used to specify a range of events as well as their relationships with each other. These models are used during work sessions and meetings to communicate important information to various role players. To help with the process, models must be presented in a graphical format using specific notations to specify concepts contained in a model and understood by everyone. (Seidewitz, 2003:27,28-29).

According to Delen and Perakath (2003: 257-268) enterprise models are used by various role-players that include users, managers, analysts and software developers focussing on various perspectives of the organisation to help with important business decisions. Delen and Perakath also list challenges of enterprise modeling that include the following (1) issues pertaining to the various tools used to support modeling, (2) problems with integrating these models, (3) the restricted focus of models (models that tend to focus on a specific domain of the enterprise i.e. data and process models), and (4) issues relating to using models to simulate and to generate code to implement systems. (Delen and Perakath, 2003: 260-262). Integration issues are also identified by (Vasconcelos et al, 2004:225-233; Soley et al, 2000:1). Another issue that enterprises have to deal with is that of the complexity of enterprise systems as identified by various authors (Delen and Perakath 2003: 257-268; Cernosek and Naiburg, 2004).

Architectural frameworks are available to address some of the challenges already discussed. Frameworks are important since it contains structural elements that could be reused by the organisation. (Marten and Robertson, 1999). Vasconcelos et al, identify the Zachman framework as one of the first enterprise architectural frameworks that adopt a holistic framework representing views of the scope, business, system and technology (Vasconcelos et al, 2004:226; Zachman,2005) Other architectural frameworks proposed by Vasconcelos et al, are a framework for Enterprise Architecture Planning (Vasconcelos et al, 2004:226-227; Spewak, 1992:13-18; The Open Group:2003). Also important is the concept of Model Driven Architecture (MDA) which is a framework resulting in code that can be generated in conjunction with models. (Uhl, 2003, Soley et al, 2000:1; Frankel et al 2003:1).

Large organisations have come to realise the benefits of using enterprise architecture frameworks to present specific views of the organisation. Some of these frameworks include the US DOD Architecture Framework and the US Federal Enterprise Architecture Framework (FEAF) as well as the Treasury Enterprise Architecture Framework (TEAF). Organisations are also using architecture frameworks as a reference framework to group, classify and document import aspects in relation to the organisation. Available reference frameworks include the ISO Reference Model for Open Distributed Processing (ISO RM-ODP) and the Zachman Framework. A description of all the frameworks is provided in “The Open Group Architecture Framework (TOGAF)” version 8.1. (The Open Group 2003: 323-334).

4.1. Complexity of Organisations.

The complexity that is part of organisations has already been identified as an issue that enterprises have to deal with. (Delen and Perakath 2003:257-258). There are many factors listed for the unsuccessful implementation of Information Technology projects but one contributing factor is the level of complexity that must be dealt with. To address the issue Lethbridge and Laganière suggest that the structure of the Information system must be understood and analysed before making any changes. (Lethbridge and Laganière, 2005:24). Complexity was also one of the issues Zachman identified when he suggested the Zachman framework. He suggested a way in which to logically define all the aspects that are part of the structure of information systems. (Zachman 1987).

4.2. Enterprise Architecture Defined.

In all the different definitions of architecture in all the different disciplines; the structure of components together with their relationships as well as principles controlling any changes are included in the description. (The Open Group, 2003:9). Definitions of other sources such as ANSI/IEEE Std 1471-2000 were also discussed in the Open Group Architecture Framework. (The Open Group, 2003:9). The definition also applies to software and hardware architectures as well as to organisations that must use the software. In the literature the word enterprise is used to indicate a specific scope of the organisation under discussion together with goals of how they add value to the organisation mission. An enterprise could include various other sub-organisations and departments. (The Open Group 2003:9).

4.3. Zachman Framework Defined.

The Zachman Framework is an enterprise architecture framework developed by J.A. Zachman and published in 1987. It draws the analogy between the building of a house and the development of an Information system. He describes a framework that could be used to address the building blocks of an enterprise architecture. The original framework by Zachman has since been extended and was published in 1992 by J.F. Sowa and J.A. Zachman. (Zachman 1987; Sowa and Zachman 1999). It describes and specifies the artefacts that are important and necessary to build successful information systems. (Martin and Robertson, 1999). An artifact can be classified as any element that is part of a functioning ICT system. It can include any element such as requirements documentation, manuals or even a software module. (Schach 2005:19). The Zachman Framework can also be considered a reference system containing a categorisation of those artefacts. (Martin and Robertson, 1999). The

Zachman framework is taxonomy of system specifications and how they fit together. (Sowa and Zachman, 1992:590-591). The Zachman Framework is also considered the *de facto* standard when specifying architectures and describing the artefacts supporting them. (The Open Group, 2003:338).

One of the strength or weakness of the Zachman framework lies in the fact that it hides the complexity of the artefacts that is necessary to build an information system. (Marten and Robertson, 1999).

Evidence that the Zachman Framework has matured is evident in the various applications and use of the Framework. The framework has been analysed and expanded by various authors. The framework is included in system analysis and development textbooks and is used by them to discuss various topics as well as to map the Zachman framework with other frameworks.(Whitten et al 2004; The Open Group 2003; Martin and Robertson, 1999; Frankel et al 2003). The Information Framework (IFW) by Evernden draws an analogy with the Zachman Framework and adds to the two (2) architecture dimensions (rows and columns) specified by the Zachman framework. The IFW also suggest that that there is a specific order in developing the dimensions as opposed to the Zachman framework that specifies that there is no order when addressing the columns-dimension. (Evernden, 1996:37-40; Sowa and Zachman 1992:599).

The Zachman Framework is a two-dimensional matrix consisting of six rows and six columns giving 36 cells that could contain possible representations of artefacts. The initial framework consisted of three columns (Data, Process and Network). (Zachman 1987:463). The columns consisted of questions or uncertainties that must be addressed. The columns were later extended to six columns including People, Time and Motivation. (Sowa and Zachman, 1992: 600-601). The six rows of the Framework contained a collection of specific functions performed by the main stakeholders that were part of the process to develop ICT systems. An analogy of the rows is depicted by Zachman of those stakeholders that are involved in the building of a house. The horisontal dimension or rows consist of a Planner, Owner, Designer, Builder and Sub-contractor. The vertical dimension is the columns also known as *focuses*. Martin and Roberson call the questions *interrogatives*. The horizontal dimension is also sometimes known as *perspectives*. (Zachman, 1987; Maartin and Roberson 1999)

The rest of the cells in the Zachman Framework contain mechanisms that put into perspective all the different roleplayers *(perspectives)* and the most important facets or characteristics *(focuses)* that must be addressed during the system development life-cycle. Each cell in the matrix contains a set of architectures.

Focus

Perspective

	C1 Data What?	C2 Function How?	C3 Network Where?	C4 People Who?	C5 Time When?	C6 Motivation Why?
R1 Planner Scope						
R2 Owner Enterprise						
R3 Designer System						
R4 Builder Technology						
R5 Sub-contractor Components						
R6 Functioning System						

Table 1: Zachman Rows and Columns.

All the artefacts must be completely or comprehensively described to achieve the maximum benefits of using the Zachman Framework. (Zachman 1998).

The following are a set of conventions or rules that generally govern the types of artefacts that are classified per cell:

- New concepts must be added to create new models in each cell. If models of diagrams are enhanced by adding more details it could also be classified as a new model. Important to note, is that if a model is re-engineered the result must be a model on a row above the present row. A quality perspective covered by the framework is that the next row must adequately describe the current row. (Sowa and Zachman 1992:603).

- One column is not more important than the other. All the columns are of similar importance. (Sowa and Zachman 1992:599).

- Each cell has a set of basic unique representations. (Sowa and Zachman 1992:600-601)

- Each row is governed by a set of unique constraints. (Sowa and Zachman 1992:601)

4.3.1. The Concepts of Zachman Primitives and Composites Defined.

The concept of primitives has also been identified by Sowa and Zachman. A primitive can be described as the smallest building block of a cell and can be used on its own. Once defined, the primitives can be combined into other more meaningful structures or diagrams. (Sowa and Zachman 1992:608; Frankel et al 2003:4). The concept of primitives is important and will be used to classify examples of artefacts in each Zachman cell.

One Zachman cell could consist of a set of primitives such as narrative descriptions, attributes and types or instances of objects which would serve the purpose to enhance the description of the cell. Once the primitives have been identified it should also be possible to store the primitives in a repository for possible future extraction for reporting purposes.

As soon as primitives of Zachman cells are related together the resulting structure are defined as a composite. This was also been identified by Sowa and Zachman when they described the integration of cells within one Zachman row in order to describe the perspectives of a specific stakeholder. (Sowa and Zachman, 1992:603). The concept of composite will be described further to show how it is possible to combine cells of different rows together and not only cells within one row as Sowa and Zachman suggest.

4.3.2. Defining Zachman Rows.

R1 Planner or Scope.

The scope or parameters where the ICT system must operate in is decided in Row 1. Concepts discussed here are of a strategic nature and one of the actions is to determine the boundaries of the organisation and how will ICT systems be used within the organisation. The external environment must also be analysed and captured. Any budget constraints must be adhered to. (Zachman 2001, Zachman 1998, Sowa and Zachman 1992:592). Work performed here, could be described as of a strategic nature. The planner view could also determine how all the components fit together.

R2 Owner or Enterprise

All the activities that are important to the business are described in Row 2. The level of obtaining data is high-level and all the business activities must eventually link to show the business value of what will be achieved if the business activity is performed. (Sowa and Zachman 1992:592). Techniques such as Business Process Modeling are important in Row 2. The perspective can show how external policies are interpreted and applied within the organisation. (Sowa and Zachman, 1992:592).

Aerts et al included business processes, the resources to support the processes as well as environmental aspects in the business architecture domain. A management sciences discipline is necessary to successfully identify and capture information. (Aerts et al, 2004:781-794)

R3 Designer or System

The level of detail specified in row 3 remains on a conceptual level and is classified as a logical level since more detail is specified in row 3 than row 2. Important to note is that the level of detail in row 3 is not yet physical. The requirements of the user are specified. (Zachman, 2001).

This row is a first step in creating application architecture. A computer science background is necessary to successfully identify and capture information on this row. (Aerts et al, 2004:781-794). System analysis and design techniques will be used effectively in Row 3. (Sowa and Zachman 1992:592). All the disciplines described in the Software Engineering field are important in Row 3.

R4 Builder or Technology

The concepts used in row 4 are inclined to be more of a physical nature together with some logical views. The physical hardware used in the system is specified. The physical system must be designed together with the connected network as well as services and devices. (Sowa and Zachman 1992:592, Zachman, 2001).

A computer systems engineering background are necessary to successfully identify and capture information on this row. (Aerts et al, 2004:781-794).

R5 Sub-contractor or Components

Row 5 would contain the physical concepts that are used implement executable code. The physical concepts can include any detailed specifications. (Sowa and Zachman 1992:592). A component is the physical piece of code or software, database or executable that is developed and used by programmers. (Schach 2005:73; Whitten et al 2004:692; Sommerville 2004:717). All the commercial-off-the- shelf (COTS) products can form part of row 5.

R6 Functioning System

The level of detail in row 6 is also of a physical nature. The actual ICT system has been created and all the concepts created are tangible. (Zachman 1987:463). It can be argued that row 6 can be ignored since it is not part of the architecture of developing an ICT system.

4.3.3. Defining Zachman Columns.

C1 Data/What?

Physical things important to the business are described in this column. These things could be all the nouns used to describe it. Examples that could be used are “Bill of materials” (Zachman, 1987:461).

C2 Function/How?

All the actions performed by the business are included in this column. The verbs used to describe the functions could be indications of all the functions performed by the organisation. The process of how important things of the business get transformed by the business. (Zachman, 1987:461)

C3 Network/Where?

All the locations or places where activities are performed are described in this column.

C4 People/Who?

The types of human resources that are needed to initiate or perform an activity are described here.

C5 Time/When?

This is an indication of when activities must be initiated, performed as well as be concluded. Scheduling and sequencing aspects should be the focus of this column. Specific time periods could also be described here.

Event modeling could be a used (Sowa and Zachman, 1992:597).

C5 (time) and C4 (people) have a close correlation with each other since the parameters that are required wherein a task must be completed indicate the amount of resources that would be necessary to complete the task. If a 24-hour availability is required, sufficient personnel would be required to address questions and issues that could arise (Sowa and Zachman 1992:597).

C6 Motivation or Why?

All the reasons of why activities are important and must be performed are indicated in this column.

4.3.4. Defining the Zachman Cells.

The functioning enterprise (Row 6) was not further elaborated by Sowa and Zachman when the original framework was extended to accommodate the additional three columns (who, when and why) (Sowa and Zachman 1992). The framework stopped at row 5, the sub contractor, row. When analysing architecture, it could be concluded that a functioning system was not part of an architecture model. Architecture has to do with the "building blocks", to build a functioning system. Architecture plays an important role when any ICT system is maintained. The building blocks used in the maintenance or evolution of systems could be the classification of the changes, for instance: any enhancements or corrections due to system failures. However the focus of this paper is not on the maintenance phase which would typically reside within row 6.

Note: During the rest of the paper any reference to the Zachman Framework and examples used would consist of 30 cells, a five row by six column matrix. Table 2 will be used when referencing to the cells.

	C1 Data What?	C2 Function How?	C3 Network Where?	C4 People Who?	C5 Time When?	C6 Motivation Why?
R1 Planner Scope	R1-C1	R1-C2	R1-C3	R1-C4	R1-C5	R1-C6
R2 Owner Enterprise	R2-C1	R2-C2	R2-C3	R2-C4	R2-C5	R2-C6
R3 Designer System	R3-C1	R3-C2	R3-C3	R3-C4	R3-C5	R3-C6
R4 Builder Technology	R4-C1	R4-C2	R4-C3	R4-C4	R4-C5	R4-C6
R5 Sub-contractor Component	R5-C1	R5-C2	R5-C3	R5-C4	R5-C5	R5-C6
R6 Functioning System						

Table 2: Zachman Index of Cells

When describing the artefacts in each cell the artefacts will be classified into nodes and links. Links cannot be alone and must be accompanied by nodes. Links will be typically the relationships between two nodes. An entity or list could be an example of a node (Sowa and Zachman 1992:593, Frankel et al 2003:2, 3). In the following pages examples of nodes and links would be provided.

R1-C1: Planner/Scope and Data/What?

A list of things that are important to the enterprise and which could be an indication of the structural components of the enterprise which are expressed by nouns (Zachman 1987:461-462).

NODE: List of important things (Sowa and Zachman 1992:600).

R1-C2: Planner/Scope and Function/How?

A list of actions that must be performed by the enterprise expressed by verbs (Zachman 1987:462). The high-level actions can be represented in a value chain representation also used by Michael Porter describing the primary and secondary activities of an organisation (Ward and Griffiths 1998:216-224).

NODE: Value Chain Primary and Secondary activities.

R1-C3: Planner/Scope and Network/Where?

A list of locations/places which could be points on a geographical map (Zachman 1992:600).

NODE: List of locations.

R1-C4: Planner/Scope and People/Who?

List of organisations that must be accommodated within the enterprise (Sowa and Zachman 1992:600).

NODE: List of organisations.

R1-C5: Planner/Scope and Time/When?

Important events imposed by external policies or events are listed (Sowa and Zachman 1992:597). Event would indicate that something important occurs on a predetermined day and time.

NODE: List of events.

R1-C6: Planner/Scope and Motivation/Why?

A list of business strategies that would ensure that objectives are met are examples of the cell (Sowa and Zachman 1992:598). A list of critical success factors could also be descriptions of examples.

NODE: List of Strategies.

R2-C1: Enterprise/Owner and Data/What?

Business entities together with relationships are described in this cell (Zachman 1987:463).

NODE: Business entities.

LINK: Business relationships.

R2-C2: Enterprise/Owner and Function/How?

Different processes of the business could be grouped together and be presented in a specific view showing how the business processes are connected (Zachman 1987:463; Sowa and Zachman 1992:600).

NODE: Business Processes

LINK: Business Resources providing inputs and receiving outputs (Frankel et al 2003:2).

R2-C3: Enterprise/Owner and Network/Where?

Any business location used by the business. It could also include facilities hosting any infrastructure. (Zachman Ebook describing the cells).

NODE: Business Locations.

LINK: Links connecting the Business Locations (Sowa and Zachman 1992:600).

R2-C4: Enterprise/Owner and People/Who?

An organisation chart describing the different responsibilities of organisations (Sowa and Zachman 1992:597).

NODE: Organisation Chart.

LINK: Work performed (Sowa and Zachman 1992:600).

R2-C5: Enterprise/Owner and Time/When?

The artefacts in this cell are “Business events” that are the result of how important external events are accommodated in the business to ensure that value is added to the organisation. Business events can also imply a certain action followed by a response within an acceptable time-frame (Whitten et al 2004:67). There is also a correlation with C4-R2 concerning “performance levels” of resources of the organisation (Sowa and Zachman 1992:597). Organisation policies will also address important business events that must be adhered to by other business units.

NODE: Business Events.

LINK: Length of time.

R2-C6: Enterprise/Owner and Motivation/Why?

A business plan containing business objectives is described in this cell (Sowa and Zachman, 1992:599).

NODE: Business Objective.

LINK: Business policies or instructions would ensure that the business objectives would be reached.

R3-C1: Designer/System and Data/What?

Data entities with attributes together with relationships are described in the cell and specified as a logical data model. (Sowa and Zachman 1992:600). Operations could be included in the logical data model.

NODE: Data entity.

LINK: Data relationship.

R3-C2: Designer/System and Function/How?

The system requirements together with an activity flow are described in this cell. Non-functional as well as functional hardware requirements are included. The logical boundaries of the system could also be captured.

NODE: Function/Activity.

LINK: Function/Activity Flow.

R3-C3: Designer/System and Network/Where?

A classification of a distributed, centralised or mobile requirement focussing on location characteristics (Frankel et al 2003:2).

NODE: Functions.

LINK: Characteristics of the communication between the functions (Frankel et al 2003:2).

R3-C4: Designer/System and People/Who?

The role that must be performed by a person is described in this view (Sowa and Zachman, 1992:597). A skills profile can be used to indicate the roles that must be adopted by the persons involved.

NODE: Role.

LINK: Interaction between the roles.

R3-C5: Designer/System and Time/When?

Important "System events" are included in this cell and indicate the duration of activities in the system as well the logical sequencing and synchronisation of events (Sowa and Zachman 1992:598).

"User response time" could be described as the moment when a user expects an answer from the system (Stallings: 2005:34-38).

NODE: System event.

LINK: Duration of events.

R3-C6: Designer/System and Motivation/Why?

Business rules are described in this cell. The business rules could be described with decision tables or decision trees. Constraints could also be specified using the Object Constraint Language.

NODE: Hierarchy of results.

LINK: Action or formula used.

R4-C1: Builder/Technology and Data/What?

Physical data model is described that would be technology dependent.

NODE: Data Segment/ Data Rows (Sowa and Zachman 1992:600).

LINK: Keys and pointers (Sowa and Zachman 1992:600).

R4-C2: Builder/Technology and Function/How?

A system design together with hardware and software components that are technology dependent. System integration and system interfaces are important.

NODE: System function.

LINK: Interfaces between the system functions.

R4-C3: Builder/Technology and Network/Where?

A "Protocol Architecture" represents a representation of the various protocols necessary to enable communication over a network (Stallings 2005:565). The "Protocol Architecture" could contain a representation such as the Open Systems Interconnection (OSI) reference model developed by the International Organisation for Standarisation (ISO). Communication mechanisms (i.e. applications, session, transport, network etc) are presented in various layers. A similar protocol, the TCP/IP protocol, could also be used (Stallings 2005:122-125).

NODE: Hardware or system software (Frankel et al 2003:2).

LINK: The way the hardware or system software are connected.

R4-C4: Builder/Technology and People/Who?

Possible Human-Machine interfaces are addressed in this view. This is where technology is used by humans or users (Sowa and Zachman 1992:597). Graphical and web-based interfaces could be included as well as voice, video and text interfaces.

NODE: User (Frankel et al 2003:2).

LINK: Screen or presentation mechanism (Frankel et al 2003:2).

R4-C5: Builder/Technology and Time/When?

"System response times" could be described as the time that it will take when a user presses "enter" and the system responds with a command (Stallings: 2005:34-38). The middleware connecting components as well as the ability to send messages between people could also be included.

NODE: Hardware function.

LINK: Hardware cycle.

R4-C6: Builder/Technology and Motivation/Why?

A list of technology standards that ICT systems must adhere to, could be described in this cell.

NODE: Standard.

LINK: Action that is performed that would comply to the standard.

R5-C1: Sub-Contractor/Component and Data/What?

Data fields used by programs that are linked to the physical data model.

NODE: Data field (Sowa and Zachman 1992:600).

LINK: Address (Sowa and Zachman 1992:600).

R5-C2: Sub-Contractor/Component and Function/How?

This cell could indicate COTS or custom-built systems using a specific programming language. (Whitten et al 2004:68). Specific menu functions are also included.

NODE: Specific program language code (Sowa and Zachman 1992:600).

LINK: Program Control Blocks (Sowa and Zachman 1992:600).

R5-C3: Sub-Contractor/Component and Network/Where?

Network topology with appropriate hubs, switches and gateways.

NODE: Specific addresses (Sowa and Zachman 1992:600).

LINK: Protocol standards (Sowa and Zachman 1992:600).

R5-C4: Sub-Contractor/Component and People/Who?

All important security aspects such as access control and authentication mechanisms are included in this cell (Sowa and Zachman 1992:597).

NODE: Security Functions.

LINK: Security Mechanisms.

R5-C5: Sub-Contractor/Component and Time/When?

Transmission speeds. An indication of transmission speeds concerning text, voice and images (Stallings: 2005:34-38).

NODE: Hardware Function.

LINK: Machine Cycle (Frankel et al 2003:2).

R5-C6: Sub-Contractor/Component and Motivation/Why?

System rules derived from the business rules and coded in a specific programming language are described in this cell. Any error messages of vendor documentation are included.

NODE: Error Message.

LINK: Supporting documentation.

Refer to Appendix B for a summary with the most important artefacts as discussed in paragraph 4.3.

4.4. Applicability of the Zachman Framework.

To illustrate the applicability and flexibility of the Zachman Framework various authors include a mapping of their analysis and findings from the Zachman framework in their articles and papers (The Open Group 2003:333-334,340-349; Noran 2003; Frankel et al 2003).

The Open Group Architecture frame performs a mapping where the TOGAF domains are compared to the Planner, Owner, Designer and Builder Rows as well as to all the six columns of the Zachman

Framework. The TOGAF domains include views concerning Business, Applications, Data and Technology. An architecture vision is also included (The Open Group 2003:340-349).

Noran performed a mapping addressing the following perspectives:

- The Generalised Enterprise Reference Architecture and Methodology (GERAM) framework or ISO 15704:2000.

- Presenting the rows of the Zachman framework as life-cycle processes.

- A set of likely modeling language to help in the presentation views of the cells. Earlier versions of the UML are presented and the applicability of the suggested UML models as well as the intended audience are taken into consideration. It would also seem that more than one UML diagram could address more than one Zachman cell. Refer to Figure 11 Comparative UML and Zachman Support (Noran). Figure 11 was adapted from the mapping that Noran did to show the various modeling language that could support the artefacts in the Zachman framework. Figure 11 only includes UML.

(Noran 2003:163-173).

The Zachman framework is mapped to models part of the Model Driven Architecture(MDA). Models part of the MDA includes the Computation-Independent Model (CIM), The Platform-Independent Model (PIM) as well as the Platform-Specific Model (PSM) (Frankel et al 2003:9). Frankel goes further and performs a UML-Zachman mapping (Frankel et al 2003:11). Refer to Figure 12 Comparative UML and Zachman support. OMG's MDA. Figure 12 was adapted from the UML-Zachman mapping that Frankel et al performed indicating UML support to create artefacts in the Zachman Framework.

As mentioned in an earlier reference to the Zachman framework, has also been included and used in System Analysis and Design textbooks to facilitate discussion of various topics (Whitten et al 2004)

4.5. Benefits of Using the Zachman Framework.

The importance of integration in the context of Enterprise Application Integration is described by Satzinger et al as the process of linking several views in order to enhance the flow of information.

ICT systems could be considered as one of the views (Satzinger et al, 2004:722-723). Vasconcelos et al also raises the importance of integration and presented a view of how architecture frameworks, the Zachman Framework and TOGAF, could support Enterprise Application Integration (Vasconcelos et al, 2004:225-233).

Integration would be achieved when all the information concepts are understood by all the levels of the organisation. These concepts must be shared and reused by all the stakeholders in order to ensure successful implementation of ICT systems (Zachman, 2001)

All the work performed to implement a system must be aligned with each other to ensure that value is added to the business or organisation. An enterprise view that support integration must be understood and communicated to all the various stakeholders of the enterprise (Zachman, 2001).

Implementing and using an architecture will ensure that work done, must be reviewed and authorised before work is started by the next team. Configuration procedures must be used in the process to ensure that the models are always “base-lined” as well as to make sure that changes could occur and would be maintained accurately. It could help with change management in the enterprise.

5. Chapter 5. Mapping RUP and UML within the Zachman Framework.

5.1. Mapping RUP within the Zachman Framework.

Iterative development has been identified as a best practice method (Cernosek and Naiburg 2004:3) during systems development and it seems that the Zachman framework could also support the technique. Martin and Robertson as well as Noran describes that each cell of the Zachman matrix include *recursively* artefacts that would contribute to a successful information system. The Zachman framework also prescribes a top-down analysis method (Martin and Robertson, 1999;Noran 2003:170). Another fact to reaffirm the use of the iterative technique is the technique supported by the Zachman Framework of decomposing models by adding more detail to those models during each subsequent iteration. The iterative development method is prescribed by RUP and included as a best practice in their methodology (IBM 2003).

It seems a logical deduction that RUP could be mapped within the Zachman Framework. It is proposed that the RUP phases could correspond to the Zachman rows since the rows represent all the stakeholders taking part in the development process. The Zachman rows are described in detail in the preceding chapter. (Chapter 4. 4.3.2 **Defining Zachman Rows.**) The following mapping is proposed in Table 3 to the Zachman rows:

Rows together with RUP stages and reasons for classification.	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**
R1 Planner. Strategic. **Inception.** Manage scope bmo actors and use cases.						
R2 Owner. Bus Process. **Inception.** Business modeling as one of the required disciplines. Determine what is important to the business and construct important business domains. Business system. Business use cases and actors, business events.						
R3 Designer. System. **Inception, Elaboration.** Determine requirements. Use cases, Use case package. Analysis class. Design package. Design subsystem. Design Class, Design model. Interfaces.						
R4 Builder. Technology. **Inception, Elaboration, Construction.** Technology specific components that are constructed as well as deployment models.						
R5 Sub-Contractor. Implementation. **Inception, Elaboration, Construction, Implementation.** Use UML implementation models. Technology specific.						
R6 Functioning System						

Table 3: The RUP within Zachman Rows.

(IBM 2003).

In the appendixes **UML Primitives. UML Composites. UML Types Used in Every Diagram.** RUP would be used as part of the reason for classifying UML elements in certain rows of the Zachman framework.

5.2. UML Primitives.

Every UML diagram (OMG UML version 1.5) was analysed and the smallest UML building blocks were identified. The building blocks were classified as UML Primitives or UML Composites according to the rules described in par 4.3.1 (**The Concepts of Zachman Primitives and Composites Defined.**) During the analyses the focus was on the UML representations, described in par 2.5 (**UML Representations.**) All the UML diagrams were decomposed into representations with names, descriptions and an example of what the representations should look like. It is important to note that the specific UML Diagram Name is in all instances part of the UML Primitive Name. It is important for reference purposes later on in the paper.

The UML primitives were linked to the corresponding Zachman columns and rows together with reasons for the classification. Refer to Appendix C, UML Primitives per UML Diagram Mapped in the Zachman Cells, for a description of the classifications per UML diagram. The following Annexes are specified within Appendix C:

- C1. Use Case Diagram.
- C2. Sequence Diagram.
- C3. Collaboration Diagram.
- C4. Activity Diagram.
- C5. State Chart Diagram.
- C6. Class Diagram.
- C7. Object Diagram.
- C8. Component Diagram.
- C9. Deployment Diagram.

The following comments are made in regard to Appendix C:

- A UML primitive could be classified in more than one row and or column for example the Actor. This would confirm the complexity of UML diagrams. Various interpretations and deliberations go into constructing any UML diagram.
- Similar UML primitives i.e actors and objects could be used by more that one diagram. This would suggest that the diagrams promote reuse of UML elements between the diagrams.

This also suggests that the different UML diagrams complement each other by using similar UML elements (actors and objects).

- Similar UML primitives are used by different role-players (rows). This would suggest that UML diagrams could contain various levels of detail and that version control would be very important to track changes of the primitives.
- Using similar RUP stages in rows suggest the incremental approach that should be adopted when ICT systems are developed. Similar primitives are used with more detail by the different role-players or rows of the Zachman framework.

5.3. UML Composites.

A similar process described in par 5.2 (**UML Primitives.**) was followed to classify UML Composites.

UML Primitives on their own do not describe the viewpoint of a stakeholder sufficiently. The value is obtained when the primitives are used together with other types of information i.e. within a matrix, model or some type of management report containing important data fields. When analysing the UML diagrams it became apparent that UML diagrams referred to more that one column during the same representation. It could also be used by more than one row as more details become known to the various stakeholders taking part in the development process. Refer to Appendix D UML Composites per UML Diagram Mapped in the Zachman Cells, for a description of the classifications per UML diagram. The following Annexes are specified within Appendix D:

- D1. Use Case Diagram.
- D2. Sequence Diagram.
- D3. Collaboration Diagram.
- D4. Activity Diagram.
- D5. State Chart Diagram.
- D6. Class Diagram.
- D7. Object Diagram.
- D8. Component Diagram.
- D9. Deployment Diagram.

The following comments are made in regard to Appendix D:

- Only a limited number of composites are described in the paper but it could be expected many more could be constructed.

- Other applications from textbooks such as descriptions for narrative specifications or using other matrixes would suggest that the UML diagrams could be extended to accommodate ease of use.
- Given the number of composites the following were concluded:
  - The complexity of UML diagrams are confirmed.
  - UML is applicable within the Zachman framework only if the UML diagrams are decomposed into simpler and basic model elements.

5.4. UML Types Used in Every Diagram.

During the analysis of the UML diagrams certain UML elements were identified that were present in every UML diagram. These elements were grouped in Appendix E, UML Types Present in all UML Diagrams, and were also grouped into Primitives and Composites, similar to Appendix C and D. The following annexes are specified within Appendix E:

- E1. UML Primitives Mapped in the Zachman Cells.
- E2. UML Composites Mapped in the Zachman Cells.

5.5. Zachman Framework with UML Primitives.

The entire set of UML primitives identified per UML diagram (refer to Appendix C) were mapped together in a Zachman-5X6-representation. Refer to Appendix F All the UML Primitives Mapped in the Zachman cells.

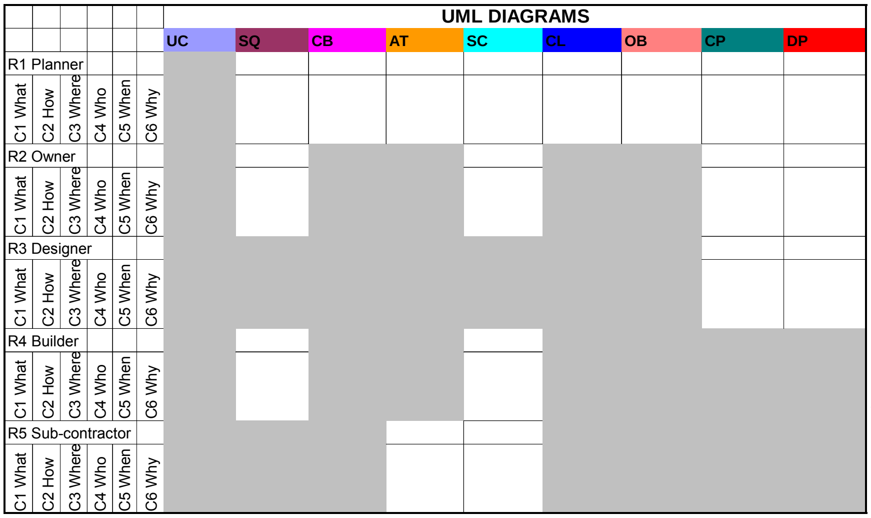
Using Appendix F it could be concluded that UML could represent elements within the Zachman Framework. This would be very subjective since various primitives could be identified per cell pending on how they were implemented and used in organisations. A more useful deduction would be to combine the UML primitives as well the UML composites to show where UML could be used within the Zachman framework.

5.6. Where is UML Used within the Zachman Framework?

To show where UML is used within the Zachman Framework the complete framework together with all the Zachman rows and columns are presented on the vertical access of Appendix G. All the UML diagrams are presented on the top horizontal axis of Appendix G. The UML diagrams are all presented with a different colour. All the UML primitives (Appendix C and E) together with the

UML Composites (Appendix D) are mapped to the Zachman framework on the vertical axis. Support is indicated by means of a tick mark in the corresponding colour. The tick mark means that a specific UML diagram could be used to provide support to the corresponding cell within the Zachman framework. Important to note is that many other UML Composites could be identified. The UML Composites identified in this paper are only a single example of such a possible combination. It is also the intention of the representation not to show any order or sequence of the Zachman cells. The level of UML support is also not indicated but one would expect that various levels could be used, from a conceptual level (row 1 and 2) up to a more detailed level when the appropriate information becomes known on row 3 to 5. Refer to Figure 10 for a comparative view of UML and Zachman support. Figure 10 is a summary view of the information described in Appendix G. The grey-shaded area indicates UML support in the Zachman rows and columns.

Figure 10 suggests that the most UML support exists on row 3, the designer level and rows 4 and 5, the builder and sub-contractor level. Some support is provided by UML diagrams on row 2, the owner level with practically no support for row 1, the Planner level.



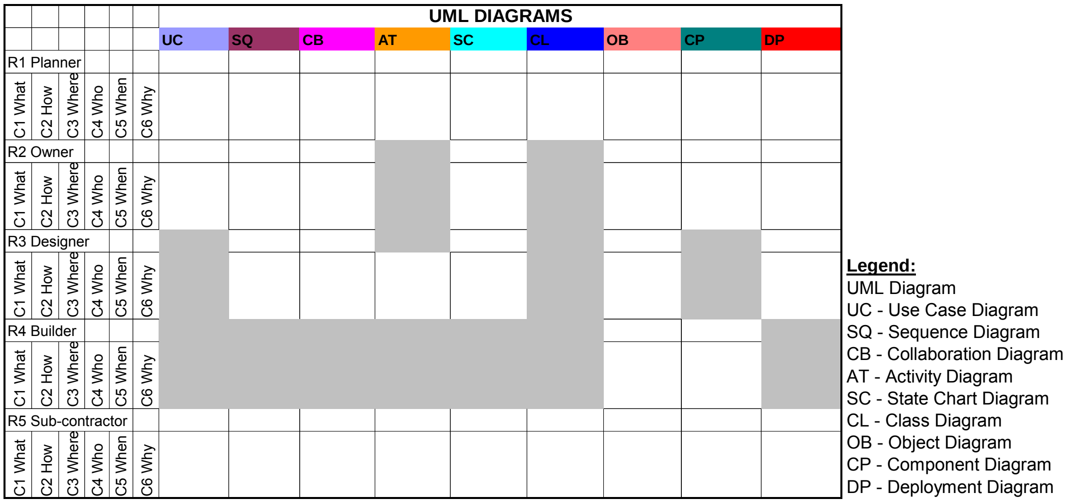
	UML DIAGRAMS								
	UC	SQ	CB	AT	SC	CL	OB	CP	DP
R1 Planner C1 What, C2 How, C3 Where, C4 Who, C5 When, C6 Why	▨								
R2 Owner C1 What, C2 How, C3 Where, C4 Who, C5 When, C6 Why	▨		▨	▨		▨	▨		
R3 Designer C1 What, C2 How, C3 Where, C4 Who, C5 When, C6 Why	▨	▨	▨	▨	▨	▨	▨		
R4 Builder C1 What, C2 How, C3 Where, C4 Who, C5 When, C6 Why	▨		▨	▨		▨	▨	▨	▨
R5 Sub-contractor C1 What, C2 How, C3 Where, C4 Who, C5 When, C6 Why	▨	▨	▨			▨	▨	▨	▨

Legend:
UML Diagram
UC - Use Case Diagram
SQ - Sequence Diagram
CB - Collaboration Diagram
AT - Activity Diagram
SC - State Chart Diagram
CL - Class Diagram
OB - Object Diagram
CP - Component Diagram
DP - Deployment Diagram

Figure 10: UML and Zachman Support.

The representation of Figure 10 has been used in the following examples of using UML within the Zachman framework. Noran as well as the White Paper on The Zachman Framework and the OMG's Model Driven Architecture represented a view of UML support within the framework. The representations of the two papers were adapted to the representation of Figure 10 with the purpose to make a comparison (Noran 2003:173; Frankel et al 2003:11). To conclude a combined representation of all three perspectives will be presented.

Noran identified a range of modeling languages that could be used to create elements within Zachman cells. Figure 11 is an adapted version of the mapping of Noran only taking into consideration the UML modeling language contained in the mapping of Noran. The grey-shaded areas indicate Zachman support. Figure 11 shows the most support that UML can provide is on row 4, the builder level, with less support on row 3, the designer level. The least support is provided on row 2, the Owner row. Noran identifies no UML support on row 1, the Planner row, and row 5 the Sub-contractor row.

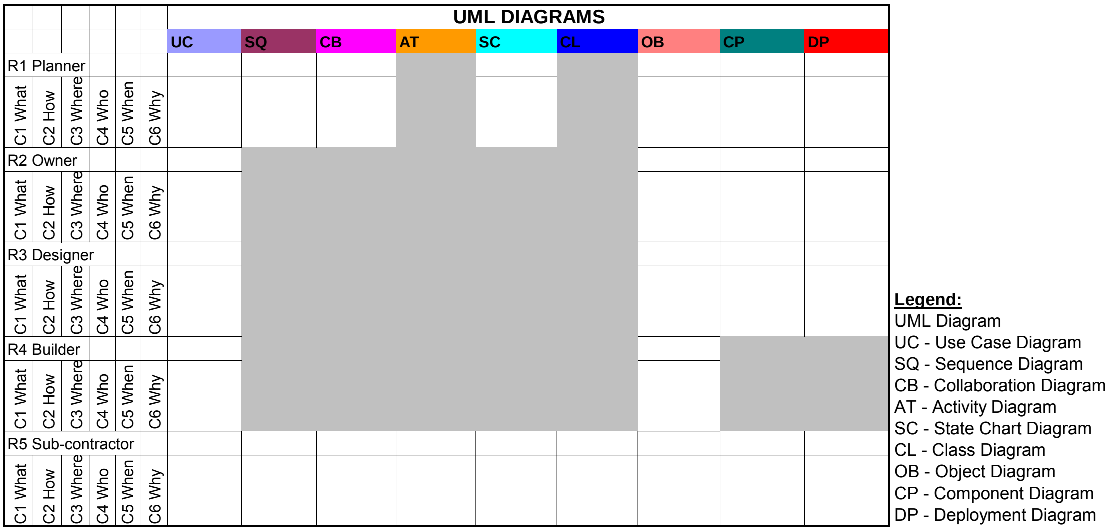


						UML DIAGRAMS								
						UC	SQ	CB	AT	SC	CL	OB	CP	DP
R1 Planner														
C1 What	C2 How	C3 Where	C4 Who	C5 When	C6 Why									
R2 Owner									▨		▨			
C1 What	C2 How	C3 Where	C4 Who	C5 When	C6 Why				▨		▨			
R3 Designer						▨			▨		▨		▨	
C1 What	C2 How	C3 Where	C4 Who	C5 When	C6 Why	▨					▨		▨	
R4 Builder						▨	▨	▨	▨	▨	▨			▨
C1 What	C2 How	C3 Where	C4 Who	C5 When	C6 Why	▨	▨	▨	▨	▨	▨			▨
R5 Sub-contractor														
C1 What	C2 How	C3 Where	C4 Who	C5 When	C6 Why									

Legend:
UML Diagram
UC - Use Case Diagram
SQ - Sequence Diagram
CB - Collaboration Diagram
AT - Activity Diagram
SC - State Chart Diagram
CL - Class Diagram
OB - Object Diagram
CP - Component Diagram
DP - Deployment Diagram

Figure 11: Comparative UML and Zachman Support. (Noran).

The next representation was adapted to represent UML support within the framework done by Frankel et al in the White Paper, The Zachman Framework and the OMG's Model Driven Architecture. Figure 12 shows the adapted UML support. The grey shaded areas indicate Zachman support. Figure 12 also shows the least support for row 5, the Sub-contractor row and the Planner row, row 1. The most support is provided for row 4, the Builder row. Frankel et al also considers Row 5 to be "out of context" (Frankel et al 2003:11).



						UML DIAGRAMS								
						UC	SQ	CB	AT	SC	CL	OB	CP	DP
R1 Planner									▨		▨			
C1 What	C2 How	C3 Where	C4 Who	C5 When	C6 Why				▨		▨			
R2 Owner							▨	▨	▨	▨	▨			
C1 What	C2 How	C3 Where	C4 Who	C5 When	C6 Why		▨	▨	▨	▨	▨			
R3 Designer							▨	▨	▨	▨	▨			
C1 What	C2 How	C3 Where	C4 Who	C5 When	C6 Why		▨	▨	▨	▨	▨			
R4 Builder							▨	▨	▨	▨	▨		▨	▨
C1 What	C2 How	C3 Where	C4 Who	C5 When	C6 Why		▨	▨	▨	▨	▨		▨	▨
R5 Sub-contractor														
C1 What	C2 How	C3 Where	C4 Who	C5 When	C6 Why									

Legend:
UML Diagram
UC - Use Case Diagram
SQ - Sequence Diagram
CB - Collaboration Diagram
AT - Activity Diagram
SC - State Chart Diagram
CL - Class Diagram
OB - Object Diagram
CP - Component Diagram
DP - Deployment Diagram

Figure 12: Comparative UML and Zachman Support. OMG's MDA.

The last representation is a combined view adapted from Noran, Frankel et al and Figure 10. Figure 13 shows the combined representation for possible UML support within the Zachman framework. The grey-shaded areas indicate Zachman support. The Sequence Diagram (SC), Collaboration Diagram (CB), State Chart Diagram (SC), Object Diagram (OB), Component Diagram (CP) as well as the Deployment Diagram (DP) provides no support on the Planner row, row 1. This can be explained since the relevant information will not be available on the Planner row. The type of information used in the planner row is very high-level conceptual information since the stakeholders on this level deal mainly with strategic information best described by lists of important concepts expressed in English. The information is also technology independent and information used in component and deployment diagrams is very much technology dependent. A similar argument could be used for the no support of the component and deployment diagrams on row 2, the Owner row.

The Activity Diagram can be used for row 1 to 4 to show the flow of activities. The reason that no support is identified for activity diagrams on row 5 could be that the flow of events is already present in the specific programming language. A similar argument could be used for the fact that State Chart Diagrams could also not be used on Row 5, the Sub-contractor row.

As a concluding comment. UML could support the Zachman framework but it is important to understand for what purpose the diagram would be used and what view would be illustrated and communicated to the enterprise by doing so.

						UML DIAGRAMS								
						UC	SQ	CB	AT	SC	CL	OB	CP	DP
R1 Planner						▨			▨		▨			
C1 What	C2 How	C3 Where	C4 Who	C5 When	C6 Why	▨			▨		▨			
R2 Owner						▨	▨	▨	▨	▨	▨	▨		
C1 What	C2 How	C3 Where	C4 Who	C5 When	C6 Why	▨	▨	▨	▨	▨	▨	▨		
R3 Designer						▨	▨	▨	▨	▨	▨	▨	▨	▨
C1 What	C2 How	C3 Where	C4 Who	C5 When	C6 Why	▨	▨	▨	▨	▨	▨	▨	▨	▨
R4 Builder						▨	▨	▨	▨	▨	▨	▨	▨	▨
C1 What	C2 How	C3 Where	C4 Who	C5 When	C6 Why	▨	▨	▨	▨	▨	▨	▨	▨	▨
R5 Sub-contractor						▨	▨	▨			▨	▨	▨	▨
C1 What	C2 How	C3 Where	C4 Who	C5 When	C6 Why	▨	▨	▨			▨	▨	▨	▨

Legend:
UML Diagram
UC - Use Case Diagram
SQ - Sequence Diagram
CB - Collaboration Diagram
AT - Activity Diagram
SC - State Chart Diagram
CL - Class Diagram
OB - Object Diagram
CP - Component Diagram
DP - Deployment Diagram

Figure 13: Combined Comparative UML and Zachman Support.

6. Chapter 6. Conclusion.

The purpose of this study was to map UML within the Zachman framework. The concluding remarks about the Zachman Framework will only address the UML primitives since the Zachman framework is a "primitive framework" containing the building blocks of IT systems. If the building blocks had been identified it would be possible to combine them into a successful ICT system. The UML primitives as well as the UML composites would be taken into consideration during the concluding remarks regarding UML support within the Zachman framework.

6.1. Zachman Columns not Addressed.

The Zachman columns are important since it is an indication of important characteristics that must be addressed when developing and classifying enterprise systems. When any Zachman columns are not addressed it means that important characteristics cannot be represented.

The Zachman Column that was the least supported by any UML diagram is the where or network column. A reason for this is that elements in this column are usually associated with elements in other columns. A location combines elements from the what-column and the who-column. Elements from the how-column plays an important role where messages are sent from a source to a destination. (Appendix C,F).

The motivation column is only supported by Use Cases by means of the narrative extensions of a Use Case Diagram. Templates can be created where narrative aspects could be addressed. (Appendix C, F).

6.2. Zachman Rows not Addressed.

The Zachman rows are the different *perspectives* or viewpoints of the types of stakeholders that is part of a systems life-cycle. When rows are not addressed it means that those viewpoints of the stakeholders can not be represented.

The Builder and Sub-Contractor rows were the least supported by UML diagrams. A reason for this is that the Builder and Sub-Contractor combines various perspectives to implement systems and since the Zachman framework is a primitive frame work, primitives on their own will be insufficient to represent their viewpoints. The planner row is also not sufficiently addressed by UML diagrams except for representations by the Use Case Diagram. (Appendix F).

6.3. Significance of UML Diagrams.

UML is the most significant when the UML building blocks (primitives) are combined together to form other UML constructs (composites). This is an important aspect since UML diagrams could also be considered composite diagrams. It is further complicated by the ability of UML to be extended to include more composite artefacts.

It would seem that the UML diagrams that offered the most support would be the Use Case and Class Diagrams. The activity diagrams offered the most support at a high-level when the least amount of detail was available, usually at the start of a development process. Sequence, Component and Deployment Diagrams provided the most support at a more technical level where systems were more dependant on the technology choices that were made. (Figure 13).

6.4. Benefits of Mapping with UML.

It must be realised by organisations that want to invest in UML as a modeling tool that the most benefit would be gained in the analysis and design of ICT systems and less support on the modeling of business processes or to model what is important to the business. The least support would be provided on the planning and conceptualising of ICT systems. (Figure 13). UML should form part of a set modeling tools used to support the entire organisation in addressing ICT issues from planning to implementation successfully.

6.5. General Conclusion.

The various gaps, i.e. the columns and rows not addressed in the Zachman framework by UML, is an indication that UML is not sufficient to support the representation of enterprise architectures and must be supported by other techniques.

6.6. Future Research.

It must be realised that UML is only one technique that can be mapped within the Zachman framework. Other techniques such as dataflow diagrams or entity relationship diagrams could also be used and mapped within the Zachman Framework. During such a mapping, gaps could also be identified that could indicate insufficient representation by the technique. The dataflow diagram, for instance, is only used by certain stakeholders and only addresses the What-column. So could various other techniques also be mapped in the Zachman framework and their suitability be tested within enterprise architecture systems.

7. References.

AAGEDAL J.O., MILOSEVIC Z., Enterprise modeling and QoS for command and control systems. *Enterprise Distributed Object Computing Workshop.* 1998. EDOC '98. Proceedings second International. 3-5 Nov 1998. Pages 88-101. [online].

AERTS A.T.M., GOOSSENAERTS J.BM., HAMMER D.K., WORTMAN J.C., Architectures in Context: on the evolution of business, application software and ICT platform architectures. Elsevier *Information & Management* 41 (2004) 781-794. [online].

ALHIR S.S., *UML in a Nutshell. A Desktop Quick Reference.* 1998.O'Reilly & Associates.

BASTOS R.,M., RUIZ D.,D.,A. Extending UML activity diagram for workflow modeling in production systems. *System Sciences.* 2002. HICSS. Proceedings of the 35th Hawai International Conference on 7-10 Jan 2002. Pages 3786-3795. [online].

BENNETT S., MCROBB S., FARMER R., *Object-Oriented Systems Analysis and Design using UML*. 2002. McGraww-Hill.

BJöRKANDER M., KOBRYN C., 2003. Architecting Systems with UML 2.0. *Software,* IEEE. Volume 20, Issue 4, July-Aug 2003. Pages 57-61. [online].

CALIO A., AUTIERO M., BUX G., Software Process Improvement by Object Technology. IEEE. *Software Engineering*. 2000. Proceedings of the 2000 International Conference on 4-11 June 200. Pages 641-647. [online].

CERNOSEK G, NAIBURG E., 2004. The Value of Modeling. A technical discussion of software modeling, *IBM USA*. [online]. Available from http://www-128.ibm.com/developerworks/rational/library/6007.html accessed 23 January 2005.

COCKBURN A., An article *Structuring Use Cases with Goals* [online] obtained from http://alistair.cockburn.us/crystal/articles/sucwg/structuringucswithgoals.htm 7/12/2004 accessed 5 June 2005.

CYSNEIROS L.,M.;PRADO LEITE J.,C.,S.; An article *Using UML to Reflect Non-Functional Requirements* [online] obtained from the portal.acm.org website. 2001.

DELEN, D., PERAKATH, C.B., August 2003. Towards a truly integrated enterprise modeling and analysis environment. *Computers in Industry*.[online]. Volume 51 Issue 3. Pages 257-268.

EVANS A., SAMMUT P., WILLANS J.S., MOORE A., MASKERI G., 2005. *Journal of Object Technology*. Vol 4 No 1 January-February 2005. Pages 165 – 181. [online]. Available from http://www.jot.fm

EVERNDEN R.,1996. The Information Framework. *IBM Systems Journal*.Vol 35 NO 1. Pages 37-68. [online].

FELDMAN D., MICALLEF J.,MULCARE D., 2003. Enterprise-Wide solutions architecting using UML. *Engineering of Computer-Based Systems*. Proceedings. 10th IEEE International Conference and Workshop on the 7-10 April 2003. Pages 191-199. [online].

FOWLER M. [online] http://martinfowler.com/bliki/UseCases.html accessed 5 June 2005.

FIRESMITH D.G., 2005. Are your requirements complete? *Journal of Object Technology*. Vol 4 No 1, January-February 2005. Pages 27 - 43. [online]. Available from http://www.jot.fm

FRANKEL D.,S., HARMON P., MUKERJI J., ODELL J., OWEN M., RIVITT P., ROSEN M., SOLEY R.,M. September 2003. Business Process Trends Whitepaper. The Zachman Framework and the OMG's Model Driven Architecture.

GRISS M.L., Architecting for large-scale systematic component reuse. IEEE. *Technology of Object-Oriented Languages.* 1998. TOOLS 26. Proceedings 3-7 Aug 1998. Pages 8-16.

IBM, 2003. IBM Rational SUMMIT Ascendant with Rational Unified Process release 8.1r. An evaluation version of the Unified Process can also be obtained from http://www-306.ibm.com/software/adwtools/rup accessed accessed 7 July 2005.

LETHBRIDGE T., C., LAGANIERE R., 2005. *Object-Oriented Software Engineering. Practical Software Development using UML and Java.* Second edition.McGraw-Hill.

MARTIN R.,ROBERTSON E., 1999. *Formalization of Multi-level Zachman Framework.* Technical Report No 522. [online]. Referenced at http://www.informatics.indiana.edu/ research/publications/0499.pdf on 2 August 2005.

NORAN O., 2003. An Analysis of the Zachman Framework for enterprise architecture from the GERAM perspective. *Annual Reviews in Control.* Volume 27, Issue 2, 2003, Pages 163-183. [online].

OLIVIER M.,S. 2004. Information Technology Research. A practical guide for Computer Science and Informatics. Second Edition 2004. Van Schaik Publishers.

OMG. 2003. OMG Unified Modeling Language Specification Version 1.5 March 2003. [online]. Available from http://www.omg.org/technology/documents/formal/uml.htm accessed February 2005.

OMG. 2004. The UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms. [online]. Available from http://www.omg.org/technology/documents/modeling_spec_catalog.htm accessed 16 January 2005.

OMG. 2005. Introduction To OMG's Unified Modeling Language ™(UML®) available from http://www.omg.org/gettingstarted/what_is_uml.htm updated 28 March 2005 and accessed 30 April 2005.

PRESSMAN R.,S., 2005. *Software Engineering A Practitioner's Approach*. Sixth Edition. McGraww-Hill International Edition.

RATIONAL STAFF. IBM 19 August 2003. *RUP Implementation Guide Part I: Recommended Strategy and typical issues and risks.* [online]. Obtained on 13 July 2005 from http://www-128.ibm.com/developerworks/rational/library/1719.html

SATZINGER J.,W., JACKSON R., B., BURD S.,D. 2004. *System Analysis and Design in a Changing World*. Third edition. Thompson Course Technology.

SCHACH S.,R., 2004. *Object oriented Analysis and Design with UML and the Unified Process.* McGrall-Hill.

SCHACH S.,R., 2005. *Object orientation & classical software engineering.*Sixth Edition. McGrall-Hill.

SEIDEWITZ, E., Sept – Oct 2003. What Models Mean. *Software IEEE*. Volume 20 Issue 5. Pages 26-32. [online]..

SEI (SOFTWARE ENGINEERING INSTITUTE) website http://www.sei.cmu.edu/str/descriptions/deda.html [online] last updated 12 May 2005 and accessed 22 May 2005.

SOLEY, R and the OMG Staff Strategy Group, 2000. *Model Driven Architecture.* White Paper [online] available from http://www.omg.org/mda/presentations.htm 00-11-05.pdf accessed 16 January 2005.

SOMMERVILLE, I., 2004. *Software engineering*. Seventh Edition. Pearson Education Limited.

SOWA J.F. , ZACHMAN J.A.. 1992. Extending and formalizing the framework for information systems architecture. *IBM Systems Journal*. Vol 31, No 3. Pages 590- 616. Information about Zachman can also be obtained as an electronic book available at www.zachmaninternational.com.

SPEWAK, S.H., 1992. *Developing a blueprint for data, applications, and technology: enterprise architecture planning*. New York: John Wiley & Sons.

STALLINGS, W., 2005. *Business Data Communications.* Fifth Edition. International Edition. Pearson Prentice Hall.

TANAKA A., NAGASE Y., KIRYU Y., NAKAI K., Applying ODP Enterprise Viewpoint Language to hospital information systems. *Enterprise Distributed Object Computing Conference.* 2001. Proceedings. Fifth IEEE International 4-7 Sept 2001 Pages 188-192. [online].

THE OPEN GROUP, 2003. *TOGAF (The Open Group Architectural Framework) Version 8.1 "Enterprise Edition"*. [online]. Available from http://www.opengroup.org/bookstore/catalog/ accessed 19 February 2005.

UHL A., 2003. Model driven architecture is ready for prime time. *Software.* IEEE. Volume 20. Issue 5. Sept-Oct 2003. Pages 70-72. [online].

VASCONCELOS, A., MIRA DA SILVA, M., FERNANDES, A., TRIBOLET, J., 2004. An Information System Architectural Framework for Enterprise Application Integration. *System Sciences,* Proceedings of the 37th Annual Hawaii International Conference on , 5-8 Jan. 2004. Pages 225-233.[online].

WANG G., CONE G., 2001. A method to reduce risks in building distributed enterprise systems. *Enterprise Distributed Object Computing Conference.* [online].Proceedings. Fifth IEEE international. 4-7 Sept 2001. Pages 164-168.

WARD J., GRIFFITHS P. 1998. Strategic Planning for Information Systems. 2nd Edition. Jon Wiley and sons Ltd.

WHITTEN J.,L., BENTLEY L.,D., DITTMAN K.,C. 2004. *Systems Analysis and Design Methods. Sixth edition.* McGraw Hill Higher Education.

ZACHMAN J.A..1987. A Framework for information systems architecture. IBM Systems Journal. Vol 26, No 3. Pages

ZACHMAN J.,A., 1998. The Physics of Knowledge Management. [online]. Available from the Zachman Institute for Framework Advancement (ZIFA) at http://www.zifa.com

ZACHMAN J.,A.,2001. *You can't "Cost Justify" Architecture*. [online] Available from the Zachman Institute for Framework Advancement (ZIFA) at http://www.zifa.com

ZACHMAN J.A., 2005. *The Zachman Framework* [online]. Available from the Zachman Institute for Framework Advancement (ZIFA) at http://www.zifa.com accessed 14 February 2005.

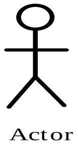
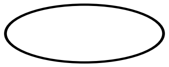
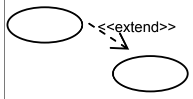
Appendix A – Abbreviations.

No	Abbreviation	Description
a	b	c
1	UC	Use Case Diagram
2	SQ	Sequence Diagram
3	CB	Collaboration Diagram
4	AT	Activity Diagram
5	SC	State Chart Diagram
6	CL	Class Diagram
7	OB	Object Diagram
8	CP	Component Diagram
9	DP	Deployment Diagram

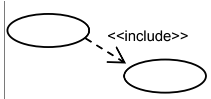
Appendix B. Examples within the Zachman Framework.

	C1 Data **What?**	**C2 Function** **How?**	**C3 Network** **Where?**	**C4 People** **Who?**	**C5 Time** **When?**	**C6 Motivation** **Why?**
R1 Planner **Scope** (Strategic, external environment, boundaries, budget constraints)	R1-C1 List of things (nouns).	R1-C2 Value Chain. (List of primary and secondary activities.)	R1-C3 A list of geographical locations.	R1-C4 Organisation list.	R1-C5 Important external and internal event list.	R1-C6 Business Strategy list. Critical success factors.
R2 Owner **Enterprise (**Business Processes, Business requirements, business Policies).	R2-C1 Business entities with relationships.	R2-C2 Business process.	R2-C3 Business locations including facilities.	R2-C4 Organisation Chart.	R2-C5 Business Events.	R2-C6 Business plan with business objectives. Business policies.
R3 Designer **System** (User Requirements, Application. Logical not physical.)	R3-C1 Data entities, attributes and operations with relationships.	R3-C2 Requirements (User and system requirements).	R3-C3 Distributed, centralized or Mobile locations.	R3-C4 Roles with responsibilities.	R3-C5 System events (sequencing and synchronisation, user responses).	R3-C6 Business rules with proposed results (decision tables & Decision trees, OCL).
R4 Builder **Technology** (Physical, hardware, network, Standards)	R4-C1 Physical data model (Technology dependant).	R4-C2 System design (hardware & software components, system interfaces).	R4-C3 Protocol architecture (OSI reference model, TCP/IP Protocol Architectures).	R4-C4 Human machine interfaces (ie graphical or voice recognition etc).	R4-C5 Transaction processing (System response time).	R4-C6 Technology standards.
R5 Sub-contractor Component (Implementation, COTS applications, Machine execution)	R5-C1 Data definitions/ tables/ segments.	R5-C2 COTS or custom built using Programming Language.	R5-C3 Network topology. (Hubs, switches, gateways).	R5-C4 Security aspects (access control, authentication mechanisms).	R5-C5 Transmission speeds (voice, text, image).	R5-C6 System rules coded in a programming language. Error messages. Vendor documentation.
R6 Functioning System						

C1 Use Case Diagram

	A	B	C	D	E	F	G
1	**Use Case Diagram (UC) Primitives**						
2				**Zachman Column**		**Zachman Row**	
3	**UC Primitive Name**	**Description**	**Representation**	**Column no**	**Reason for classification**	**Row no**	**Reason for classification**
4	UC Actor	An actor operates within the boundaries or environment of the system. It can be a person or a system. It can also be a role or set of responsibilities. An actor interacts with the system and initiates actions or receives output		C4 Who?or C1 What? or C5 When	If the actor is a person it can be classified in the C5 People column. If the actor is a system it can be classified in the C1 What? column. If the actor represents time (time actor) the actor can be classified in the C5 When? column.	R1 - R5	Actors can be identified by every stakeholder. R1 determine the scope, R2 identifies business actors and use cases, R3 determine the boundaries of the requirement, R4 technology specific actors (i.e. system administrators) and R5 the actors can be part of the system testing and implementation (all the roles that are accommodated in the system such as users and authorisation entities.
5	UC Use Case	The use case describes the interactions.		C2 How? & C2 How?	The use case indicates a specific action.	R1-5	On R1 use cases can be identified that will help determining the scope. On R2 business use cases can help determining business domains. On R3 Analysis use cases can help determining the system functionality that must be tested and implemented.R4 technology use cases indicating the functions that must be done for technology solutions such as video conferencing. R5 test scenarios.
6	UC Extend	Relationship between use cases. The relationship indicates that functionality of one use case is enhanced by another use case.		C2 How? & C2 How?	Use case (How) and extend relationship is also (How) since it enhances functionality of a use case.	Row 3	Enhanced functionality will only be determined in design models done in RUP phase elaboration.

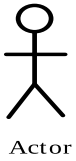
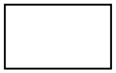
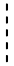
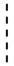
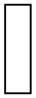
C1 Use Case Diagram

	A	B	C	D	E	F	G
1	**Use Case Diagram (UC) Primitives**						
2				**Zachman Column**		**Zachman Row**	
3	**UC Primitive Name**	**Description**	**Representation**	**Column no**	**Reason for classification**	**Row no**	**Reason for classification**
7	UC Include	Relationship between use cases indicating that the interactions are also included in the other use case. (OMG 2003:498)		C2 How? & C2 How?	Use case (How) and include relationship is also (How) since it includes the interactions of a use case.	Row 3	Includes detailed functionality that would only become clear in design models done in RUP stage elaboration.
8	UC Source where information was obtained.	Text attribute. Can be used in a Use Case Template.	Label	C3 Where?	The attribute indicates the location where the information was obtained.	R1-3	The information would be made available by any of these stakeholders.
9	UC Interested stakeholder	Text attribute. Can be used in a Use Case Template.	Label	C4 Who?	The stakeholder would be other interested persons or organisations	R1-4	The information would be made available by any of these stakeholders. R4 technology stakeholders.
10	UC Precondition	Text attribute. Describes a constraint of the system. What must exist before the system can run. Can be used in a Use Case Template.	Label	C6 Why?	A constraint suggests that it could be a business rule.	R2-5	The constraint would be indicated by a business requirement.R4 used during technology specific solutions. R5 used during test cases.
11	UC Trigger event	Text attribute. Events that initiates use case functions are described. Can be used in a Use Case Template.	Label	C5 When?	Indication of when an action starts.	R2-5	The business process (R2) would determine it or it could be derived from a business process (R3). R4 used during technology specific solutions. R5 used during test cases.
12	UC Course of Events	Text attribute. A description of the sequence of activities performed. Can be used in a Use Case Template.	Label	C2 How?	Describes functions.	R3-5	User Requirements would determine the sequence of events. R4 used during technology specific solutions. R5 used during test cases.
13	UC Alternative Courses	Text attribute. Can be seen as a variation when a decision point is reached. Use cases do not contain decision points. Can be used in a Use Case Template.	Label	C2 How?	Describes functions.	R3-5	User Requirements would determine the sequence of events. R4 used during technology specific solutions. R5 used during test cases.

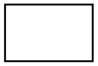
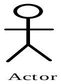
C1 Use Case Diagram

	A	B	C	D	E	F	G
1	**Use Case Diagram (UC) Primitives**						
2					**Zachman Column**		**Zachman Row**
3	**UC Primitive Name**	**Description**	**Representation**	**Column no**	**Reason for classification**	**Row no**	**Reason for classification**
14	UC Post Condition	Text attribute. Constraint after the use case has been successfully completed. Can be used in a Use Case Template.	Label	C6 Why?	A constraint suggests that it can be a business rule.	R2-5	Could indicate how subsystems within applications were linked together. System designs could indicate this.
15	UC Assumptions	Text attribute. Similar to preconditions. Can be used in a Use Case Template.	Label	C6 Why?	Can be seen as a constraint that is a business rule.	R2-5	Certain assumptions were necessary to indicate system boundaries. R4 used during technology specific solutions. R5 used during test cases.
16	UC Business Rules	policies and procedures that directs the business and also any systems. Can be used in a Use Case Template.	Label	C2 How?	Rules providing statements of constraints	R3	Analysis and design models done during describing requirements.
17	UC Use Case Description	Text attribute. Can be used in a Use Case Template.	Label	C1 What?	Data element. Summary description of the functionality that is described in the use case.	R1-5	Use Case descriptions may be described in various levels of detail . R4 used during technology specific solutions. R5 used during test cases.
18	UC Conclusion	Text attribute. Describing the value that a actor receives when the use case has been performed. Can be used in a Use Case Template.	Label	C6 Why?	The value to the organisation is described.	R1-5	Conclusions could be seen as a description as the end state that cal be obtained from user requirements.
19	UC Rank	Digit attribute. Show the importance of the use case. Can be used in a Use-Case-Ranking&-Priority-Matrix	Label	C2 How?	The rank would indicate how important the functionality is.	R3-5	The user or system requirement should determine the importance of the use case. R4 used during technology specific solutions. R5 used during test cases.
20	UC Priority	Digit attribute. It can be an indication when the use case must be developed. Can be used in a Use-Case-Ranking&-Priority-Matrix	Label	C5 When?	The attribute indicates when the use case should be developed.	R3-5	During requirement analysis an indication of the development priority should be obtained. R4 used during technology specific solutions. R5 used during test cases.

C2 Sequence Diagram

	A	B	C	D	E	F	G
1	**Sequence Diagram (SQ) Primitives**						
2					**Zachman Column**		**Zachman Row**
3	**SQ Primitive Name**	**Description**	**Representation**	**Column no**	**Reason for classification**	**Row no**	**Reason for classification**
4	SQ Actors	A person that initiates a specific action in a SQ.		C4 Who?	Person must initiate the action.	R3	Analysis & Design diagrams. RUP stage elaboration.
5	SQ Objects	Anything that requires that data must be stored about it.		C1 What?	Objects represent things	R3, R5	Analysis & Design diagrams. RUP stage elaboration. R5 physical database names.
6	SQ Actor Life Lines	Represents time of an actor.		C5 When?	The messages that an actor initiates could also be seen as an actor that initiates a system event.	R3	Analysis & Design diagrams. RUP stage elaboration.
7	SQ Object Life Lines	Indicates how long the object is busy performing messages.		C5 When?	Messages passed amongst objects could also be classified as system events.	R3	Analysis & Design diagrams. RUP stage elaboration.
8	SQ Activation Bar	Indicates the duration of messages.		C5 When?	An activation bar could indicate the duration of system events.	R3	Analysis & Design diagrams. RUP stage elaboration.
9	SQ Timing Label	A time label is printed on the left-hand side of activation bars.(OMG 2003:503).	{receive time < 10 sec}	C5 When?	The formula indicates that the message must be sent/received within a specific period of time.	R3, R5	Analysis & Design diagrams. RUP stage elaboration. Information is only created during system analysis & design. R5 statements could be expressed in a programming language.
10	SQ Destruction Label	Label. Indicating that a message sequence 'dies'.	X	C2 How?	A destruction can be seen as a system functionality	R3	Analysis & Design diagrams. RUP stage elaboration.

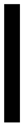
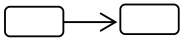
C3 Collaboration Diagram

	A	B	C	D	E	F	G
1	**Collaboration Diagram (CB) Primitives**						
2					**Zachman Column**		**Zachman Row**
3	**CB Primitive Name**	**Description**	**Representation**	**Column no**	**Reason for classification**	**Row no**	**Reason for classification**
4	CB Collaboration	Instance of a role that is performed. (OMG 2003:515)		C1 What?	A role is an example of an important thing that a system must keep data about.	R 2, 3, 4. 5	The different organisation roles are important to the business (Row 2). The different skill profiles (also a role) could be determined during system analysis and design. (Row 3). On Row 4 - Technology participants must interact with each other. Row 5 Participants taking part in test and implementation activities would also interact with each other.
5	CB Actor	An actor that initiate a set of collaborations.		C4 Who? or C1 What?	If an actor is a person (Who) or an actor could also be a system (What).	R 2,3,4,5	Actors could be identified during process design (Row 2) or during system analysis (Row 3). Technology and infrastructure actors (Row 4). Test actors (Row 5).
6	CB Arrow Label	Interaction description	Text expression	C1 What?	An example of data	R 2,3,4,5	Interaction descriptions could be identified by the business (Row 2) or by system analysis (Row 3). On Row 4 - Technology participants must interact with each other. Row 5 Participants taking part in test and implementation activities would also interact with each other.

C4. Activity Diagram.

	A	B	C	D	E	F	G
1	**Activity Diagram (AT) Primitives**						
2					**Zachman Column**		**Zachman Row**
3	**AT Primitive Name**	**Description**	**Representation**	**Column no**	**Reason for classification**	**Row no**	**Reason for classification**
4	AT Start activity	Start of a sequence of activities	●	C5 When	Indicate at what point (When) a group of activities can start.	R2,3,4	Could be indicated by a business process (R2 RUP stage inception) or system process (R3 RUP stage elaboration).R4 Technology Design to indicate the flow of activities (RUP stage construction)
5	AT End Activity	End of a sequence of activities	◉	C5 When	Indicate at what point (When) a group of activities stops.	R2,3,4	Could be indicated by a business process (R2 RUP stage inception) or system process (R3 RUP stage elaboration). R4 Technology Design to indicate the flow of activities (RUP stage construction)
6	AT Activity	Activity or process	▢	C2 How?	Indicate a function that must be performed.	R2,3,4	Could be indicated by a business process (R2 RUP stage inception) or system process (R3 RUP stage elaboration). R4 Technology Design to indicate the flow of activities (RUP stage construction)
7	AT Decision Block	Indicate that a decision is necessary in the process.	◇	C2 How?	Decision is a type of function.	R2,3,4	Could be indicated by a business process (R2 RUP stage inception) or system process (R3 RUP stage elaboration). R4 Technology Design to indicate the flow of activities (RUP stage construction)

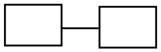
C4. Activity Diagram.

	A	B	C	D	E	F	G
1	**Activity Diagram (AT) Primitives**						
2					**Zachman Column**		**Zachman Row**
3	**AT Primitive Name**	**Description**	**Representation**	**Column no**	**Reason for classification**	**Row no**	**Reason for classification**
8	AT Synchronisation Bars	Indication that a set of different activities can occur at the same time.		C5 When?	Indicates a point in time that different sets of activities can occur at the same time.	R2,3,4	Could be indicated by a business process (R2 RUP stage inception) or system process (R3 RUP stage elaboration). R4 Technology Design to indicate the flow of activities (RUP Stage construction)
9	AT Activity Flow	Process flow between two functions.		C2 How? & C2 How?	The flow is an indication of how data is communicated between functions (How).	R 2,3,4	Business functions are classified in Row 2 and system functions in Row 3 during system analysis and design (RUP stage elaboration). Row 4 Physical design (RUP stage construction).

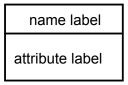
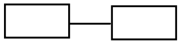
C5. State Chart Diagram.

	A	B	C	D	E	F	G
1	**State Chart Diagram (SC) Primitives**						
2					**Zachman Column**		**Zachman Row**
3	**SC Primitive Name**	**Description**	**Representation**	**Column no**	**Reason for classification**	**Row no**	**Reason for classification**
4	SC Initial State	Entry label when a group of states starts	●	C5 When?	Indicates when an group of states will starts.	R 3	Information would be modelled in system design (RUP stage elaboration). Requirements determination.
5	SC End State	Final state label.	◉	C5 When?	Indicates when a group of states end.	R 3	Information would be modelled in system design (RUP stage elaboration). Requirements determination.
6	SC State	A state can be a specific condition that an object in in at a specific moment in time. (OMG 2003:573).	Active	C1 What?	A state describes a specific data value of an object at a specific time.	R 3	Information would be modelled in system design (RUP stage elaboration). Requirements determination.
7	SC Sub-state	A decomposed state. (OMG 2003:540)	Active	C1 What?	More descriptive of a state.	R 3	Information would be modelled in system design (RUP stage elaboration). Requirements determination.
8	SC Synchronisation bars	Indicates that different paths of states can occur.	▮	C5 When?	A synchronisation occurs when different states must be possible at the same time.	R 3	Information would be modelled in system design (RUP stage elaboration). Requirements determination.

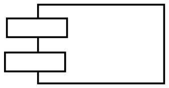
C6. Class Diagram.

	A	B	C	D	E	F	G
1	**Class Diagram (CL) Primitives**						
2					**Zachman Column**		**Zachman Row**
3	**CL Primitive Name**	**Description**	**Representation**	**Column no**	**Reason for classification**	**Row no**	**Reason for classification**
4	CL Name Label	Name of the class	objectname:class name	C1 What?	Describe things.	R2,3,4,5	Would be indicated during business process modeling (R2 RUP stage inception) or system analysis and design (R3 RUP stage elaboration). Row 4 used during physical database design (RUP stage construction). Row 5 specific database calls.
5	CL Attributes	Descriptive data values	text strings	C1 What?	Specifies a list of text attributes.	R 3,4,5	Would be indicated during system analysis and design (R3 RUP stage elaboration). Row 4 used during physical database design (RUP stage construction). Row 5 specific database calls.
6	CL Operations	Expressions indicating operations	text strings	C2 How?	Describes how functions will be performed (i.e. system behaviour).	R 3,4,5	Would be indicated during system analysis and design (R3 RUP stage elaboration). Row 4 used during physical database design (RUP stage construction). Roe 5 specific database calls.
7	CL Relationship	Indicates that their exist a relationship between objects		C1 What?	Enhances the primitive description by showing that there is a relationship between objects	Row 2,3,4,5	Relationships would be used at every row except row 1 since the planner only identifies lists of important things

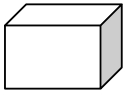
C7. Object Diagram.

	A	B	C	D	E	F	G
1	**Object Diagram (OB) Primitives**						
2					**Zachman Column**		**Zachman Row**
3	**OB Primitive Name**	**Description**	**Representation**	**Column no**	**Reason for classification**	**Row no**	**Reason for classification**
4	OB Name	Data element specifying the object name. (object name : class name).	Text label	C1 What?	Descriptive text name.	Row 2,3,4,5	Objects could be used to describe what things are important to the business (Row2). During system analysis and design object instances could be used to describe difficult classes. Row 4 physical database design. Row 5 specific database calls.
5	OB Attribute	Descriptive list of text attributes that must be stored. (attribute name : type = value)	Text attribute list	C1 What?	Descriptive nouns.	Row 3,4,5	During system analysis and design, attributes would be used to describe the important data elements that must be stored in a system. Such detailed information could be determined during requirements gathering work sessions. Row 4 physical attributes. Row 5 specific database calls.
6	OB Object	A object is an instance of a class and could include (although not necessary) attributes. (OMG 2003:464).		C1 What? & C1 What?	Name (What) and attributes (What) serves to enhance the description of an object.	Row 2,3,4,5	Objects without attributes could be identified during business processes (Row 2). It could be used to describe complicated classes during analysis and design models (Row 3 RUP stage elaboration. Row 4 is the physical design. Row 5 specific database calls.
7	OB Relationship	Indicates that there exists a relationship between objects		C1 What?	Enhances the primitive description by showing that there is a relationship between objects	Row 2,3,4,5	Relationships could be used at every row except row 1 since the planner only identifies lists of important things

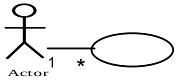
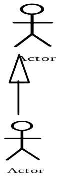
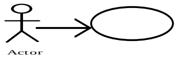
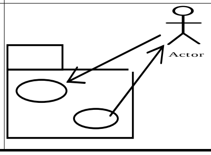
C8. Component Diagram.

	A	B	C	D	E	F	G
1	**Component Diagram (CP) Primitives**						
2				**Zachman Column**		**Zachman Row**	
3	**CP Primitive Name**	**Description**	**Representation**	**Column no**	**Reason for classification**	**Row no**	**Reason for classification**
4	CP Component	Software component that could be implemented. OMG 2003:574. Examples are programming languages.		C1 What?	If the definition of data could also accommodate things it could be classified in the C1 What? column.	Row 4,5	A software component is technology specific, that is implemented on a hardware component. Information would become available in RUP stages Construction and implementation.
5	CP Component Name	Name of component. Component attribute.	Label	C1 What?	Attribute that enhances the description of a component.	Row 4,5	A software component is technology specific, that is implemented on a hardware component. Information would become available in RUP stages Construction and implementation.
6	CP Component Type	Type of the component. Component attribute.	Label	C1 What?	Attribute that enhances the description of a component.	Row 4,5	A software component is technology specific, that is implemented on a hardware component. Information would become available in RUP stages Construction and implementation.
7	CP Component Instance	Component Name : Component Type. Component attribute	Label	C1 What?	Attribute that enhances the description of a component.	Row 4,5	A software component is technology specific, that is implemented on a hardware component. Information would become available in RUP stages Construction and implementation.

C9. Deployment Diagram.

	A	B	C	D	E	F	G
1	**Deployment Diagram (DP) Primitives**						
2					**Zachman Column**		**Zachman Row**
3	**DP Primitive Name**	**Description**	**Representation**	**Column no**	**Reason for classification**	**Row no**	**Reason for classification**
4	DP Node	A node depicts a piece of hardware.		C1 What?	If the definition of the data column is extended to include other things, a node could be classified as a thing.	Row 4,5	Technology dependent. Would be identified during RUP construction and implementation phases.
5	DP Name	Name of the hardware component.	Label	C1 What?	Deployment name can be an attribute that describes the DP Node	Row 4,5	Technology dependent. Would be identified during RUP construction and implementation phases.
6	DP Type	Classification Type.	Label	C1 What?	Deployment type can be an attribute that describes the DP Node	Row 4,5	Technology dependent. Would be identified during RUP construction and implementation phases.
7	DP Node Instance	Contains the node name and node type. (Name:type).	Label (Name:Type)	C1 What?	DP Node instance	Row 4,5	Technology dependent. Would be identified during RUP construction and implementation phases.

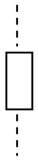
D1. Use Case Diagram.

	A	B	C	D	E	F	G
1	**Use Case (UC) Composites**						
2					**Zachman Column/s**		**Zachman Row/s**
3	**UC Composite Name**	**Description**	**Representation**	**Related Column no**	**Reason for classification**	**Related Row no**	**Reason for classification**
4	UC Multiplicity	Cardinality type of relationship between an actor and a use case. (2003:499).		C2 How? & C4 Who?	Indication of how many instances of use cases (How) and actors (Who) participates in a system.	Row 2,3	The information could be identified in business use cases(row 2 RUP stage inception) as well as analysis and design use cases (row 3 RUP stage elaboration).
5	UC Generalisation	Relationship between actors.		C2 How? & C4 Who?	Indicates how actors relate to each other.	Row 3	Detailed relationships would only be clear in analysis and design models. (Row 3 RUP stage elaboration).
6	UC Association	Relationship between actors and use cases. (OMG 2003:497).		C2 How? & C4 Who?	Indicates that certain actors (who) make use of specific use cases (How).	Row 2,3	The relationship could be identified in business (Row 2 RUP stage inception) and analysis models (Row 3 RUP stage elaboration).
7	UC Subsystem	A subsystem can contain a group of Use Cases. A sybsystem shows a specific behaviour of a system.		C1 What? & C2 How? & C4 Who?	The specific group can be classified as What and the Use cases as How. The actor that participates is classified as C4 Who	Row 3	The classification using subsystems would be done in the analysis and design models (RUP stage elaboration).

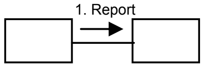
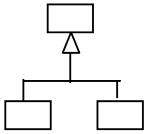
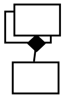
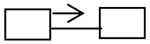
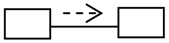
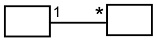
D2. Sequence Diagram.

	A	B	C	D	E	F	G
1	**Sequence Diagram (SQ) Composites**						
2					**Zachman Column/s**		**Zachman Row/s**
3	**SQ Composite Name**	**Description**	**Representation**	**Related Column no**	**Reason for classification**	**Related Row no**	**Reason for classification**
4	SQ Message	Horisontal perspective. Text that describes what is communicated (information flow). The text can contain a sequence nr describing when the message must be performed. Can be seen as a general message (asynchronous message) that requires a response	1. Message ──▶	C2 How? & C1 What? & C5 When?	The message describes an action, how data is passed between objects (what) or between actors (who) and objects. The sequence number describes when the message must be sent.	R3,R5	Analysis & Design diagrams (RUP stage elaboration). R5 The text label may contain expressions in a specific programming language. (RUP stage construction).
5	SQ Asynchronous Message	Another type of message indicating that feedback or a response are not required.	1. Message ──>	C2 How? & C1 What? & C5 When?	The message describes an action, how data is passed between objects (what) or between actors (who) and objects. The sequence number describes when the message must be sent.	R3,R5	Analysis & Design diagrams (RUP stage elaboration). R5 The text label may contain expressions in a specific programming language.
6	SQ Return Message	Another type of message indicating a response that was requested.	2. Return message - - - - - - ->	C2 How? & C1 What? & C5 When?	The message describes an action, how data is passed between objects (what) or between actors (who) and objects. The sequence number describes when the message must be sent.	R3,R5	Analysis & Design diagrams (RUP stage elaboration).R5 The text label may contain expressions in a specific programming language. (RUP stage construction).
7	SQ Recursive Message	Another type of message where data is passed	↩	C2 How? & C1 What? & C5 When?	The message describes an action, how data is passed between objects (what) or between actors (who) and objects. The sequence number describes when the message must be sent.	R3,R5	Analysis & Design diagrams (RUP stage elaboration). R5 The text label may contain expressions in a specific programming language. (RUP stage construction).

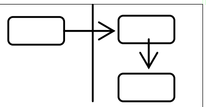
D2. Sequence Diagram.

	A	B	C	D	E	F	G
1	**Sequence Diagram (SQ) Composites**						
2				**Zachman Column/s**		**Zachman Row/s**	
3	**SQ Composite Name**	**Description**	**Representation**	**Related Column no**	**Reason for classification**	**Related Row no**	**Reason for classification**
8	SQ Vertical perspective	A vertical perspective containing life-lines and activation bars.		C5 When? & C1 What? & C4 Who?	Shows how time influence the sending of messages from sources. The source could be an actor (Who) or an Object (What)	R3	Analysis & Design diagrams (RUP stage elaboration).

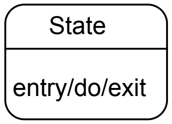
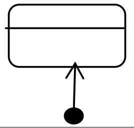
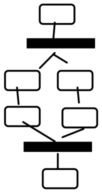
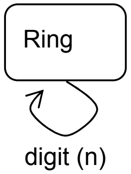
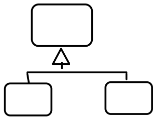
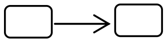
D3. Collaboration Diagram.

	A	B	C	D	E	F	G
1	**Collaboration Diagram (CB) Composites**						
2					**Zachman Column/s**		**Zachman Row/s**
3	**CB Composite Name**	**Description**	**Representation**	**Related Column no**	**Reason for classification**	**Related Row no**	**Reason for classification**
4	CB Interaction	Operations that is performed between two roles together with sequence numbers. (OMG 2003:523)		C1 What? & C4 Who? & C3 Where? & C2 How?	Roles could be people (Who) or objects (What). The sequence is (Where) and the flow (How)	R2,3,4,5	The different organisation roles are important to the business (Row 2). The different skill profiles (also a role) could be determined during system analysis and design. (Row 3).
5	CB Generalisation	Indicates the relationships between collaboration roles. OMG (2003:522)		C1 What? & C4 Who? & C2 How?	Roles could be people (Who) or objects (What). The relationship is how they are related.	R2,3,4,5	The relationship between the different roles can be done by the business (R 2) or during system analysis and design.
6	CB Composition	Composition relationship between collaboration roles. (OMG 2003:528)		C1 What? & C4 Who? & C2 How?	Roles could be people (Who) or objects (What). The relationship is how they are related.	R2,3,4,5	The relationship between the different roles can be done by the business (R 2) or during system analysis and design.
7	CB Asynchronous Interaction	Type of CL interaction. An operation is performed the collaboration role continuous with other work without waiting for an reply. OMG 2003:530		C1 What? & C4 Who? & C2 How? & C3 Where?	Roles could be people (Who) or objects (What). The flow is how the interaction are performed and the direction is where the interaction or message is sent.	R 3,4,5	Message information would only become available during system analysis and design (RUP stage elaboration).
8	CB Return Interaction	Type of CL interaction, a return interaction.		C1 What? & C4 Who? & C2 How? & C3 Where?	Roles could be people (Who) or objects (What). The flow is how the interaction are performed and the direction is where the interaction or message is sent.	R 3,4,5	Message information would only become available during system analysis and design (RUP stage elaboration).
9	CB Multiplicity	Relationship between two collaboration roles.		C1 What? & C2 How? & C4 Who?	Roles could be people (Who) or objects (What). The relationship is how they are related.	R2,3,4,5	The relationship between the different roles can be don by the business (R 2) or during system analysis and design.

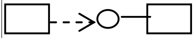
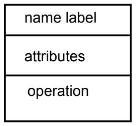
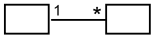
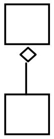
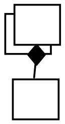
D4. Activity Diagram.

	A	B	C	D	E	F	G
1	**Activity Diagram (AT) Composites**						
2					**Zachman Column/s**		**Zachman Row/s**
3	**AT Composite Name**	**Description**	**Representation**	**Related Column no**	**Reason for classification**	**Related Row no**	**Reason for classification**
4	AT Swimlines	Indicates responsibilities of what process is performed by who		C1 What? & C4 Who? & C2 How?	Functions (What) and flows (How) are classified per responsibility (Who).	R 2,3,4	Business functions are classified in Row 2 and system functions in Row 3 during system analysis and design (RUP stage elaboration). Row 4 physical design (RUP Stage construction)
5	Activity-location matrix (ALM)	Matrix with Xs indicating what activity are performed at what location. (Satzinger 2004:233).	Text table	C2 How? & C5 Where?	Activities are classified as (How) and locations as (Where).	R 2,3,4	Business functions are classified in Row 2 and system functions in Row 3 during system analysis and design (RUP stage elaboration). Row 4 Physical design (RUP stage construction).
6	Location diagram LD	A geographical map indicating the places where the system would be used. (Satzinger 2004:232).UML stereotype.		C1 What? & C5 Where?	Systems are classified as (What) and locations or places as (Where).	R 2,4	During business process analysis (R 2) important places could be indicated on a geographical map. Row 4 physical design (RUP stage construction).
7	AD Data entry action (ALM)	CRUD (Create, Read, Update, Delete) indicating the data-action of the activity per data element. (Satzinger 2004:234).	Text table	C1 What? & C2 How?	Data elements are classified as (What) and the data-entry actions as (How).	R 3	During system analysis and design it will be determined what functions are performed on data elements.

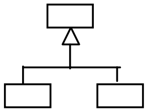
D5. State Chart Diagram.

	A	B	C	D	E	F	G
1	**State Chart Diagram (SC) Composites**						
2					**Zachman Column/s**		**Zachman Row/s**
3	**SC Composite Name**	**Description**	**Representation**	**Related Column no**	**Reason for classification**	**Related Row no**	**Reason for classification**
4	SC State (nr 2)	Describes a state as well an internal actions (entry/do/exit) that must be performed if the object is that state. (OMG 2003:538).		C1 What? & C2 How? & C5 When?	An action (How) is performed when an object is in a specific state (What).	Row 3	Information (states with action) would be available to be modelled in system design (RUP stage elaboration).
5	SC Interface	An interface is an external function that causes an specific state. OMG (2003:573)		C1 What? & C2 How? & C1 What?	An interface (What) causes a specific state (What) and action (How).	Row 3	Information would be modelled in system design (RUP stage elaboration)
6	SC Concurrent States	Describes different paths of states.OMG (2003:547).		C5 When? & C1 What?	Concurrent actions would start at a specific point indicated by a synchronisation bar (When), concurrent states (What) is possible at the same time.	Row 3	Information would be modelled in system design (RUP stage elaboration). Requirements determination.
7	SC Sequential sub-state	Indicates a recursive state. It will continue for an amount of time. OMG (2003:541)		C5 When? & C1 What?	The amount of time (When) is described together with the state (What).	Row 3	The information would be available during system design. (Requirements determination).
8	SC Event	A set of events that causes changes in states of objects. (OMG 2003:542).		C1 What? & C2 How?	An event in this context is a group of states (C1) and how they relate to each other.	Row 3	The information would be available during system design. Requirements determination.
9	SC Transition	Indicates a change in state		C1 What? & C2 How?	The flow (How) between states (What) is modelled.	Row 3	The information would be available during system design. Requirements determination.

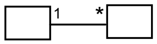
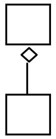
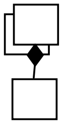
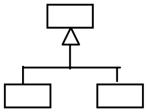
D6. Class Diagram.

	A	B	C	D	E	F	G
1	**Class Diagram (CL) Composites**						
2					**Zachman Column/s**		**Zachman Row/s**
3	**CL Composite Name**	**Description**	**Representation**	**Related Column no**	**Reason for classification**	**Related Row no**	**Reason for classification**
4	CL Interface	External operation of a class. An interface does not contain any attributes, states or relationships. (OMG 2003:451,452).		C1 What? & C2 How?	Classes are C1 What indicating the name and attributes. C2 is the external operations.	Row 4,5	Used during Row 4, physical design. Row 5 is the specific programming language.
5	CL Class	Something that contains static (attribute) and behaviour (operations) elements. The attributes and operations need not be displayed.		C1 What? & C1 What?, C6 Why?, C2 How?	The name and attributes relates to the What column and the operations are statements could be derived from business rules i.e. formulas (C6 Why) or it could indicate how certain attributes are calculated (C2 How).	Row 2,3,4,5	Identified during analysis and design models (RUP stage elaboration). Classes without attributes and operations could be used during business modeling (Row 2). Row 4 used during physical database design (RUP stage construction). Row 5 specific database calls.
6	CL Multiplicity	Cardinality-type of relationship between classes.		C1 What? & C2 How?	Indicate how classes (What) relate with each other.	Row 2,3,4	Used during business (Row 2) as well as analysis and design models (Row 3). Row 4 used during physical database design (RUP stage construction)
7	CL Aggregation	A consists-of relationship between objects. (OMG 2003:124).		C1 What? & C2 How?	Indicate how classes (What) relate with each other.	Row 2,3,4	Used during business (Row 2) as well as analysis and design models (Row 3). Row 4 used during physical database design (RUP stage construction)
8	CL Composition	Relationship between classes. More precise aggregation relationship between classes. OMG 2003:467)		C1 What? & C2 How?	Indicate how classes (What) relate with each other.	Row 2,3,4	Used during business (Row 2) as well as analysis and design models (Row 3). Row 4 used during physical database design (RUP stage construction)

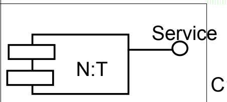
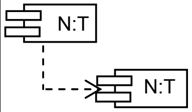
D6. Class Diagram.

	A	B	C	D	E	F	G
1	**Class Diagram (CL) Composites**						
2				**Zachman Column/s**		**Zachman Row/s**	
3	**CL Composite Name**	**Description**	**Representation**	**Related Column no**	**Reason for classification**	**Related Row no**	**Reason for classification**
9	CL Generalisation	Relationship, parent-child or superclass-class, relationship between classes. OMG (2003:718).		C1 What? & C2 How?	Indicate how classes (What) relate with each other.	Row 2,3,4	Used during business (Row 2) as well as analysis and design models (Row 3). Row 4 used during physical database design (RUP stage construction)

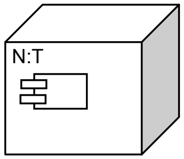
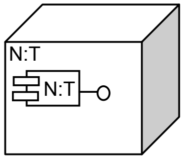
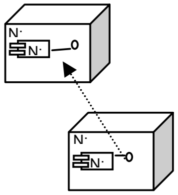
D7. Object Diagram.

	A	B	C	D	E	F	G
1	**Object Diagram (OB) Composites**						
2				**Zachman Column/s**		**Zachman Row/s**	
3	**OB Composite Name**	**Description**	**Representation**	**Related Column no**	**Reason for classification**	**Related Row no**	**Reason for classification**
4	OB Multiplicity	Cardinality-type of relationship between objects. (OMG 2003:107)		C1 What? & C2 How?	Indicates how objects (What) relate with each other.	Row 2,3,4	Could be used during business modeling (Row 2) and analysis & design models (Row 3). Row 4 is the physical design.
5	OB Aggregation	A consists-of relationship between objects. (OMG 2003:124).		C1 What? & C2 How?	Indicates how objects (What) relate with each other.	Row 2,3,4	Could be used during business modeling (Row 2) and analysis & design models (Row 3). Row 4 is the physical design.
6	OB Composition	Relationship between objects. More precise aggregation relationship between objects. OMG 2003:467)		C1 What? & C2 How?	Indicates how objects (What) relate with each other.	Row 2,3,4	Could be used during business modeling (Row 2) and analysis & design models (Row 3). Row 4 is the physical design.
7	OB Generalisation	Relationship, parent-child or superclass-class, relationship between objects. OMG (2003:718).		C1 What? & C2 How?	Indicates how objects (What) relate with each other.	Row 2,3,4	Could be used during business modeling (Row 2) and analysis & design models (Row 3). Row 4 is the physical design.

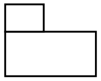
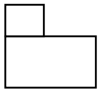
D8. Component Diagram.

	A	B	C	D	E	F	G
1	**Component Diagram (CP) Composites**						
2				**Zachman Column/s**		**Zachman Row/s**	
3	**CP Composite Name**	**Description**	**Representation**	**Related Column no**	**Reason for classification**	**Related Row no**	**Reason for classification**
4	CP Interface	Services provided by the component.		C1 What? & C3 Where? & C2 How?	An interface is located between components what) and could be an indication of possible available functionality.	Row 4,5	Interfaces are technology dependent between components. Work would be performed during RUP stages Construction and implementation.
5	CP Dependency	Dependencies may exist between software components.		C2 How? & C1 What?	A dependency is a type of function (How) and the components are classified as what.	Row 4,5	Dependencies are technology dependent. Work will be performed during RUP stages Construction and implementation.

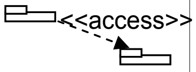
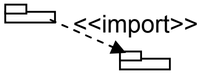
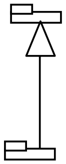
D9. Deployment Diagram.

	A	B	C	D	E	F	G
1	**Deployment Diagram (DP) Composites**						
2					**Zachman Column/s**		**Zachman Row/s**
3	**DP Composite Name**	**Description**	**Representation**	**Related Column no**	**Reason for classification**	**Related Row no**	**Reason for classification**
4	DP Node	Piece of hardware describing a element with processing capability. A node could also be a person. The node could contain component instances. (OMG 2003:573).		C1 What? & C1 What? or C4 Who?	Nodes and components are classified as What. They are things. The person is classified in C4.	Row 4,5	Technology dependent. Work will be performed during RUP stages Construction and implementation.
5	DP Interface	Services provided by the component of a node		C1 What? & C2 How?	The components and nodes could be classified as What to indicate things and the functionality contained by the interfaces are how. Interfaces could also be classified as what.	Row 4,5	Technology dependent. Work will be performed during RUP stages Construction and implementation.
6	DP Communication path	Flow indicated by an association relationship		C2 How? & C3 Where?	The flow and arrow could indicate where the flow starts and ends. The flow could also indicate how the nodes communicate with each other.	Row 4,5	Technology dependent. Work will be performed during RUP stages Construction and implementation.

E1. UML Primitives

UML primitives used in every diagram						
				Zachman Column		**Zachman Row**
UML Model Primitive	**Description**	**Representation**	**Column no**	**Reason for classification**	**Row no**	**Reason for classification**
Package	A package is a general grouping of any UML model element together.		C1 What?	The package indicates a grouping of modeling element.	R1 - R5	Packages could be used to classify models created by any stakeholder.
Subsystem	A subsystem is similar to a package but it can contain interfaces and operations between subsystems. The UML diagrams where it is used is use case and state chart diagrams. (OMG 2003:419).		C1 What?	The package indicates a grouping of modeling elements.	R1 - R5	Packages could be used to classify models created by any stakeholder.

E2. UML Composites.

UML Composites used in every UML diagram						
			Zachman Column/s		**Zachman Row/s**	
UML Model Composite	**Description**	**Representation**	**Related Column no**	**Reason for classification**	**Related Row no**	**Reason for classification**
Package access line	The access line with a label indicates that the contents of one package is referenced by another package without modifying it.		C1 - What? & C2 How?	The package indicates a specific grouping of models and the label together with the line indicates how the models can be referenced. The access label indicates the contents may me referenced without changing it. OMG 2003:463)	R1 - R5	Packages could be used to classify models created by any stakeholder.
Package import line	The import line with a label indicates that the contents of one package is referenced and updated by another package.		C1 - What? & C2 How?	The import label indicates that permission was granted to change the contents of the package. OMG 2003:464)	R1 - R5	Packages could be used to classify models created by any stakeholder.
Package generalisation line	A generalisation relationship indicates a parent-child relationship between two packages. The point indicates the parent. The child inherits the contents of the parent or the parent consists of the contents in the child package. (OMG 2003:486).		C1 - What? & C2 How?	The relationship indicates that there was a relationship between the packages (what) and also describes how they were related i.e. the parent consists of the contents in the child package.	R1 - R5	Packages could be used to classify models created by any stakeholder.

Appendix F. All the UML Primitives Mapped in the Zachman Cells.

Perspectives(rows)	**What? (Data)**	**How? (Function)**	**Where? (Network)**	**Who? (People)**	**When? (Time)**	**Why? (Motivation)**
Focusses(columns)	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**
R1 Planner	UC Actor, UC Description	UC Use Case	UC Source	UC Actor, UC Interested Stakeholder,	UC Actor	UC Conclusion
R2 Owner	UC Actor UC Description OB Name, OB Rel CB Collaboration, CB Actor, CB Arrow Label CL Name Label, CL Rel Package, Subsystem	UC Use Case AT Activity, AT Decision Block, AT Act Flow	UC Source	UC Actor UC Interested Stakeholder, CB Actor	UC Actor, UC Trigger Event AT Start Activity, AT End Activity, AT Synchronisation Bars	UC Precondition, UC Post Condition, UC Assumptions, UC Conclusion
R3 Designer	UC Actor UC Description OB Name, OB Attribute, OB Rel CB Collaboration, CB Arrow Label, CB Actor SC State, SC Sub State CL Name Label, CL Attributes, CL Rel SQ Objects Package, Subsystem	UC Use Case, UC Course of Events, UC Alt Courses, UC Buss Rules, UC Rank AT Activity, AT Decision Block, AT Act Flow CL Operations SQ Destruction Label	UC Source j	UC Actor UC Interested Stakeholder, CB Actor SQ Actor	UC Actor, UC Trigger Event, UC Priority SC Init State, SC End State, SC Synchronisation Bar AT Start Activity, AT End Activity , AT Synchronisation Bars SQ Actor Life Line, SQ Object Life Line, SQ Activation Bar, SQ Tim lab	UC Precondition, UC Post Condition, UC Assumptions, UC Conclusion
R4 Builder	UC Actor UC Description DP Node, DP Name, DP Type, DP Node Instance CP Component, CP Component Name, CP Component Type, CP Component Instance OB Name, OB Attribute, OB Rel CB Collaboration, CB Actor, CB Arrow Label CL Name Label, CL Attributes, CL Rel Package, Subsystem	UC Use Case, UC Course of Events, UC Alt Courses, UC Rank AT Activity, AT Decision Block, AT Act flow CL Operations		UC Actor UC Interested Stakeholder, CB Actor	UC Actor, UC Trigger Event, UC Priority AT Start Activity, AT End Activity, AT Synchronisation Bars	UC Precondition, UC Post Condition, UC Assumptions, UC Conclusion

R5 Sub-Contractor	UC Actor UC Description DP Node, DP Name, DP Type, DP Node Instance CP Component, CP Component Name, CP Component Type, CP Component Instance OB Name, OB Attribute, OB Rel CB Collaboration, CB Actor, CB Arrow Label CL Name Label, CL Attributes, CL Rel SQ Objects Package, Subsystem	UC Use Case, UC Course of Events, UC Alt Courses, UC Rank CL Operations		UC Actor CB Actor	UC Actor, UC Trigger Event, UC Priority SQ Timing Label, SQ Objects	UC Precondition, UC Post Condition, UC Assumptions, UC Conclusion
6 Functioning System						

Legend:

- UC Use Case Diagram
- SQ Sequence Diagram
- CB Collaboration Diagram
- AT Activity Diagram
- SC State Chart Diagram
- CL Class Diagram
- OB Object Diagram
- CP Component Diagram
- DP Deployment Diagram

Appendix G. Comparitive UML and Zachman Support (Detailed)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1							UC	SQ	CB	AT	SC	CL	OB	CP	DP
2	R1 Planner														
3															
4	C1 What?	C2 How?	C3 Where?	C4 Who?	C5 When?	C6 Why?									
5	√	√	√	√	√	√	UC Stereotype (Source,Interested Stakeholders & Conclusion), UC Actor, UC Use Case Description, Package (access, import and rel), Subsystem.								
6	R2 Owner														
7	C1 What?	C2 How?	C3 Where?	C4 Who?	C5 When?	C6 Why?									
8	√ √ √ √ √	√ √ √ √ √	√ √ √	√ √	√ √	√ √	UC Stereotype (Precondition, Trigger Event, Source, Int Stakeholders, Description, Conclusion, Assumptions), Use Cases, UC Actors,Relationships between the actor and Use Cases, Package (access, import and rel), Subsystem		Collaboration, Actor, Arrow Label, Interaction, Relationships, Package (access, import and rel), Subsystem	Activity start & end, Activity, Decision, Synchronisation Bar, Activity flow, Swimlines, Activity-Location-Matrix, Location Diagram, Package (access, import and rel), Subsystem		Class Name, Class, Relationships, Multiplicity, Package (access, import and rel), Subsystem	Object Name, Object, Relationships, Multiplicity, Package (access, import and rel), Subsystem		

Appendix G. Comparitive UML and Zachman Support (Detailed)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1							UC	SQ	CB	AT	SC	CL	OB	CP	DP
9	R3 Designer														
10	C1 What?	C2 How?	Where?	C4 Who?	C5 When?	C6 Why?									
11	√ √ √ √ √ √ √	√ √ √ √ √ √ √	√ √ √	√ √ √	√ √ √ √	√ √	Analysis Use Case, UC Actor, Use Case Stereotype (Source, Int Stakeholder, Precondition,Trigger event, Course of evemts, Alt courses, Post Condition, Assumptions, Bus rules, Use Case description, Conclusion), UC Rank/Prio matrix, UC Dependancy Diagram, UC relationships,Cardinality (Actor & UC), UC Sub system, UC include and extend, Package (access, import and rel)	Actor, Objects, Actor Life-lines, Object Life-lines, Activation Bar, Timing Constraint, Destruction Label, Message (Asynchronous, return, recursive), Vertical SQ Perspective, Package (access, import and rel), Subsystem	Collaboration, Actor, Arrow Label, Interaction, Relationships, Package (access, import and rel), Subsystem	Activity start & end, Activity, Decision, Synchronisation Bar, Activity flow, Swimlines, Activity-Location-Matrix, Data entry action, Package (access, import and rel), Subsystem	Initial & End State, State & Sub-state, Synchronisation Bar, Event, Transition, Internal state actions, Interface, Subsystem, Concurrent States, Sequential sub-states, Package (access, import and rel)	Class Name, Class, Attributes, Operations, Relationships, Multiplicity, Package (access, import and rel), Subsystem	Object Name, Object, Attributes, Relationships, Multiplicity, Package (access, import and rel), Subsystem		

Appendix G. Comparitive UML and Zachman Support (Detailed)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1							UC	SQ	CB	AT	SC	CL	OB	CP	DP
12	R4 Builder														
13	C1 What?	C2 How?	C3 Where?	C4 Who?	C5 When?	C6 Why?									
14	√ √ √ √ √ √ √	√ √ √ √ √ √ √	√ √ √ √	√ √ √	√	√	Technology Actors, Technology Use Cases, Use Case Stereotype (Int Stakeholder, Precondition, Trigger Event, Course of Events, Alt Courses, Post Condition, Assumptions, Use Case Description, Conclusion), UC Rank/Prio matrix, Subsyst, Package(access, import and rel)		Collaboration, Actor, Arrow Label, Interaction, Relationships	Activity start & end, Activity, Decision, Synchronisation Bar, Activity flow, Swimlines, Activity-Location-Matrix, Stereotype (Location Diagram), Package (access, import and rel), Subsystem		Class Name, Class, Attributes, Operations, Relationships, Multiplicity, Aggregation, Composition, Interface, Package (access, import and rel), Subsystem	Object Name, Object, Attributes, Relationships, Multiplicity, Package (access, import and rel), Subsystem	Software Component, Component Name, Type, Instance, Interfaces, Dependancies, Package (access, import and rel), Subsystem.	Hardware Node, Node Name & Type, Node Instance, Interfaces, Communication Paths, Package (access, import and rel), Subsystem
15	R5 Sub-Contractor														
16	C1 What?	C2 How? C3	Where?	C4 Who?	C5 When?	C6 Why?									
17	√ √ √ √ √ √ √	√ √ √ √ √ √	√ √ √	√ √ √	√ √ √	√	Implementation Actor, Test Use Cases, Stereotype (Precondition, Trigger Event, Course of Events, Alt Courses, Post Condition, Assumptions, Use Case Description, Conclusion), UC Rank/Prio matrix, Package (access, import and rel), Subsystem.	Objects, Timing Label, , Package (access, import and rel), Subsystem.	Collaboration, Actor, Arrow Label, Interaction, Relationships			Class Name, Class, Attributes, Operations, Interface	Object Name, Object, Attributes, Package (access, import and rel), Subsystem	Software Component, Component Name, Type, Instance, Interfaces, Dependencies, Package (access, import and rel), Subsystem..	Hardware Node, Node Name & Type, Node Instance, Interfaces, Communication Paths, Package (access, import and rel), Subsystem