Chapter 7 - Evaluating the prototype

7.1 Evaluating DSS

Evaluation of a DSS should be built into the development process. It should begin before any technical phases (analysis, design, development, testing and implementation) and should continue beyond the life of the DSS. It is the control mechanism for the entire iterative design procedure (Sprague & Carlson 1982). The evaluation process keeps the cost and effort of the DSS in line with its value. With constant evaluation, the system can die when the need for it is over or it proves not to be valuable. DSS evaluation can also help to quantify the impact of decision-making processes on organisational goals. Sprague & Carlson (1982) consider what to measure, how to measure it and presents a general model for evaluation. DSS evaluations should be considered as planned experiments designed to test one or more hypotheses.

Table 7-1 Examples of measures for DSS Evaluation (Sprague & Carlson 1982)

Productivity measures
1. Time to reach a decision
2. Cost of making a decision
3. Result of the decision
4. Cost of implementing the decision

Process Measures
1. Number of alternatives examined
2. Number of analysis done
3. Number of participants in the decision-making
4. Time horizon of the decision
5. Amount of data used
6. Time spent in each phase of the decision-making
7. Time lines of the decision

Perception measures
1. Control of the decision-making process
2. Usefulness of the DSS
3. Ease of use
4. Understanding of the problem
5. Ease of "selling" the decision
6. Conviction that the decision is correct.

Product measures
1. Response time
2. Availability
3. Mean time to failure
4. Development costs
5. Operating costs
6. Maintenance costs
7. Education costs
8. Data acquisition costs

Sprague & Carlson (1982) divide the possible evaluation measurements into four categories:

- Productivity measures to evaluate impact of the DSS on decisions
- Process measures to evaluate the impact of the DSS on decision-making
- Perception measures the evaluate the impact of the DSS on decision-makers, and
- Product measures that evaluate the technical merit of the DSS

See Table 7-1 (p126) for examples of measures of DSS evaluation. The initiating DSS' impact can be evaluated or the target system (the decision, the organisation, the product on which the impact is) can be measured (Sprague & Carlson 1982). Sprague & Carlson 1982 compare eight methods to measure and evaluate the DSS. These methods are not mutually exclusive and claim not to be system and hypothesis independent. Each is a systematic method of evaluating DSS impact.

7.1.1 Event logging

Events may indicate the DSS impact: log events before and after implementation of the DSS to gain information on actions, opinions, newspaper articles, items from memos, dates etc. Judgement is required in selecting the events to be recorded, and the set of events is not pre-definable. The recording may be on a continuous or before-and-after basis. The evaluation is simply an analysis of recorded events. This method is most useful when:

- Quantitative measures cannot be used
- Time series of effects is of interest, and/or
- Multiple effects are being considered

7.1.2 Attitude Survey

This method attempts to measure opinions based on a questionnaire issued to individuals. A large body of psychological and behavioural research can be used. For the best results, the questionnaire should be administered at various intervals during the development and use of the DSS and the results compared. An attitude survey is most useful when:

- Target system consists of people, and
- Perception measures are being used

7.1.3 Cognitive testing

Cognitive testing utilises methods developed by social psychologists and is a variation of the attitude survey. It is based on explicit theories of behaviour and more formal evaluation procedures. Behaviour patterns, preferences, processes or understanding of the DSS is evaluated. The results of the tests are often difficult to interpret. Cognitive testing will have to be done by experts.

7.1.4 Rating and weighing

This is a highly structured method for a composite numerical evaluation. This methodology involves the development of a set of parameters such as accuracy, timeliness and usability. Individuals rate each

parameter evaluating the relative importance of each parameter by assigning a specific parameter to a certain score or weight by using a scale of one to 10. The score of each individual may be averaged or totalled. When analysing the results the evaluator should consider that the numbers attached to the parameters are not precise and may not be accurate. If the ratings and weights are reliable, this method is the easiest to interpret. If not, the results will be misleading.

7.1.5 System Measurement

The performance of the target system is measured in an attempt to quantify the effect of the DSS.
If the target system is the DSS, the evaluation should be product measures for the DSS. The measurements may be collected automatically by the DSS or other sensors, through questionnaires, interviews or observations, or extracted from documents. The method of evaluation is usually a before-and-after comparison of measurements.

The analysis often utilises statistical techniques; data collection; data sampling; and the results are usually easy to interpret. This method unfortunately has a narrow range of applicability because it requires that the characteristics of the target system have quantified performance measures. System measurement is the most reliable evaluation method.

7.1.6 System Analysis

This technique involves a formalised, qualitative technique for describing the impact of the target system on multiple aspects. This evaluation involves the technique used in the analysis preceding the development of any system and may involve description of the system in terms of procedures, information flows, data stored, personnel activities, data used in report and decisions depending on the impact being investigated. Tools that are often used are interviews, document reviews, observation of data collection techniques, flow charts, organisational charts, input-output matrices and decision tables. The evaluation may contain a description or the comparison of descriptions.

This method involves both quantitative and qualitative measures. It is the best method to evaluate the target system's impact on procedures and organisational structures. The biggest advantage is that the baseline can be gathered during the analysis that precedes the DSS design. The results are usually difficult to interpret because this method does not provide a numerical measure of the system impact. An example of this technique may be to document the requirements for data used in a report and compare it with the content and retrieval capability of the DSS.

7.1.7 Cost-Benefit Analysis

This method utilises some of the data collection techniques of the system measurement and the system analysis methods. This method is more often used in feasibility studies for DSS than for evaluation of the DSS. Comparing the ratios before and after the DSS implementation can enhance this analysis. Cost-benefit impact evaluation will have the most meaningful results for managers, but may be particularly difficult to compile because the benefits and costs of a DSS may be socially and not

quantifiable. The benefits of the system may appreciate rather than depreciate over time because of the accumulation of data and experience, and because of the difficulty to allocate computer systems costs among users. Because of these problems, the method may reduce to a comparison of direct costs. Cost-benefit analysis is the only way to evaluate economic impact and it should not be ignored or limited to a comparison of direct costs and benefits.

7.1.8 Value Analysis

Keen (as referenced by Sprague & Carlson 1982) proposes a value analysis evaluation technique. It is an approach similar to the cost-benefit analysis but based on benefits first and costs second. It is the benefits that are of importance to decision-makers. Calculations of the costs are not necessary if the benefits are not acceptable. The method attempts to reduce risk by requiring prototyping as part of the evaluation method. Prototypes are relatively low-risk and an inexpensive way to obtain relatively accurate evaluation data. The method evaluated DSS as a research-and development (R&D) effort rather than a capital investment. An R&D evaluation tends to encourage innovation rather than the return on investment.

Keen (as referenced by Sprague & Carlson 1982) describes value analysis as a series of four steps:

- Establish the operational list of benefits the DSS must achieve to be acceptable
- Establish the maximum cost that one is willing to pay to achieve the benefits
- Develop the prototype DSS, and
- Assess the benefits and the costs.

The advantage of this method is that it is simple and integrated with prototyping. It may not include all the relevant measures and is a less rigorous method than the cost-benefit or system analysis techniques. Value analysis seems very close to the intuitive approach that many managers use to evaluate DSS.

7.1.9 Combining methods

The choice of the evaluation method will depend on the DSS and the impacts investigated. The evaluator and the environment too, have an impact on the evaluation method chosen. A combination of evaluation methods will result in a better evaluation because of this effect and the fact that some problems exist with all evaluation methods. System measurement, system analysis and cost-benefit analysis use similar data collection techniques and can therefore easily be combined.

7.2 ES quality and validation

ES users start as novices that do not know much about the knowledge domain. They do not know what the system can deliver (Raggad & Gargano 1999). Users become loyal because of the value they receive continually learning from the ES. One of the reasons ES fail is that ES owners' measurements, analyses and maintenance stress user satisfaction and fails to emphasise value creation (Raggad & Gargano 1999). Value creation is the key to ES success. For an ES to create value for the user continually, it has to evolve by having new knowledge added to the knowledge base regularly. The

novice user will learn the rules of the ES with time. Ragged & Gargano (1999) prove by induction that the end-user will become so literate that invoking the ES for future sessions will become infeasible. The users will only become more dependent on the system if the system learns with a higher learning rate than the end-users.

7.2.1 Knowledge and knowledge base validation

Knowledge validation refers to the checking of the accuracy of the expert system. This is achieved by extensive testing of the quality of knowledge as well as the changes in the environment that inappropriate past expert rules.

The value of the ES depends on the validity of its knowledge base. The validity or the knowledge base depends on the consistency of the reasoning scheme of the inference engine (Raggad & Gargano 1999). For the ES to be useful, it must have knowledge depth. Knowledge depth is the ability to extend existing knowledge and infer new knowledge. Contrary to the DSS-user, the ES-user is usually not familiar with the knowledge domain and this creates a risk using the ES. The ES-user who lacks a great deal of background in the problem domain hopes that the system contains the knowledge needed to solve the problem. If the entire knowledge domain of the ES cannot be represented, the defection risk is greater since the probability of missing information is high.

When the ES generates a message that may be anticipated by the decision-maker, there is no added value of information to the decision, and invoking the ES is not justified. Recommendation anticipation is measured by the difference between the value of the decision given by the ES' recommendations and the value of the decision based on prior evidence (Raggad & Gargano 1999).

7.2.2 Knowledge base verification

The power and effectiveness of the ES is equal to the quality of the knowledge it contains. To ensure quality of knowledge the knowledge base needs to be verified. Logical verification of rules can detect many problems in the knowledge base. The potential problems can be grouped into (Klein & Methlie):

- Consistency problems: caused by redundant rules, conflicting rules, subsumed rules, unnecessary if conditions and circular rules. , and
- Completeness problems: caused by missing rules because of unreferenced attribute values, missing combinations of attribute values, control faults and gaps in the inference chains.

The above logical problems may not necessarily cause reasoning faults. Klein & Methlie (1995) summarise various verification tests that verify rules in a knowledge base. Some of the techniques that verify the knowledge base for consistency and correctness are given below.

- **Consistency checking techniques**

Redundant rules:

Two rules are redundant if they succeed in the same situation and have the same conclusions. The condition parts of the rules are equivalent, and one or more of the conclusions of the two rules are the same. Redundant rules do not necessarily cause logical problems. If stored in adequate parts of the knowledge base, it may even affect performance positively. Redundant rules may however create maintenance problems. One of the rules may be maintained when the other is left unchanged.

Conflicting rules:

Two rules are conflicting if they succeed in the same situation but with conflicting solutions. Rules may be different, having equivalent conditions but different conclusions and yet do not conflict at all.

Subsumed rules:

One rule is subsumed by another, if the two rules have the same conclusions, but applies additional constraints on the situation on which it will succeed. Whenever the one rule succeeds, the other one also succeeds.

Unnecessary IF conditions:

Two rules contain unnecessary IF conditions if the rules have the same conclusions, and the IF condition of the one rule is in conflict with an IF condition in the other rule and all other IF conditions are equivalent. , and

Circular Rules:

A set of rules is circular if the chaining of the rules in the set forms a cycle. The system will move into an infinite loop. A dependency chart may help in the detecting of circular rule chains. The circular rule chain may not always be direct and apparent. An algorithm can easily be constructed to detect any implicit circular patterns among rules.

- **Techniques checking for completeness**

Missing rules:

Rules may be missing because of:

- Unreferenced attribute values, and/or
- Missing combinations of condition attribute values

Illegal attribute values:

If a rule refers to an attribute value that is not in the set of legal values, an error occurs, and

Dead ends:

A dead end occurs when a conclusion of a rule does not match the goal of an IF condition of another rule. Dead ends may be caused by terminology errors e.g. where synonyms are used.

♦ Validation

A knowledge base may be logically correct without being valid. Validation has to do with how well a model or measurement conforms to what is being modelled or measured. It tests if the system is a true representation of reality (Klein & Methlie1995).

Three types of faults may be encountered during the evaluation process:

- Factual faults: An assertion does not correctly represent the fact
- Inferential faults: A rule does not correctly represent the domain knowledge. The result is an incorrect conclusion drawn by the system. , and
- Control faults: The rules are correct, but have undesirable control behaviour

The validation is a three-step procedure:

- Run the program or problems
- Identify the faults, and
- Modify the program (Rules and control strategies)

Validity refers to relevance, meaningfulness and correctness. Three types of validations exist: content, construct and criterion related validations. **Content validation** judges each item for its presumed relevance to the property being measured. **Construct validation** refers to the validation of the models: the knowledge base, the reasoning strategies and the analytical relationships. Construct validation is dependant on the way a knowledge base is built. If the models are built around hypothesised relationships, each relationship can be validated. If built on a theory such as the theory of problem solving, constructive validation can be done by the expert's judgements of the constructs established. One such way is by a ***rule trace*** approach. The system is run on a specific problem, the system behaviour traced as the rules are tried, rejected or fired.

Criterion related validity is studied by comparing test scores with one or more criteria or measurements of the attribute to measure. Criterion-related validity is concerned with the predictive ability of the system. Criterion-related validation can be performed ***empirically*** by measuring the system's success. Statistics on overall performance may be easy, but to identify the right and wrong behaviour of individual rules is difficult.

▪ Rule traces

Running rule traces on a problem causes a search tree to develop. The rule trace is the path through this tree. It is an account of rules that have been tried, those that did not succeed and those that did. Two types of rule trace analysis can be done: the intuitive and the analytical. In the intuitive approach, a rule trace is presented to the expert who will comment on the conclusions and the reasoning. There is no explicit external reference to which the system's performance can be compared. The reference is in the expert's head. There is no systematic and controlled way of correcting the identified faults.

Changes to the program may affect other faults in other cases, cases that might have been handled well previously.

Using a more precise conceptual framework and formal procedures may result in a formal analysis to identify and correct faults. A critical point in this analysis is to determine the rules and the sequence in which they should have fired. This trace is called the ideal trace. The ideal trace can be taken as input or be determined by using problem solving and inference techniques. The ideal trace is compared with the actual trace of the rules to locate the first point at which the trace differs. The comparison determines the rule that should have fired and could not do so because it was insufficiently or incorrectly constrained or missing (Bundy et al 1985, as referenced by Klein & Methlie 1995).

It is sufficient to concentrate on errors of commission. Correcting these errors will eventually correct errors of omission. If the rule of commission is corrected, it will cause the faulty rule to not fire any more. If any other errors of commission exist, they too will be corrected to cause the most preferred rule to fire. Several modification techniques associated with rule trace comparisons exist:

- Reordering of rules to correct the control faults
- Adding extra conditions to the antecedent of a rule (also called specialisation or discrimination), and
- Concept learning techniques

- **Empirical performance validation**

This approach tests the predictive ability of the system by comparing its conclusions with one or more external variables or criteria. One criterion to use is to test the expert conclusions on a set of test cases and empirically state how well the system's conclusions match with these. The critical point is to have enough representative coverage for validation – to gather enough cases for typical decision outcomes. Special validation techniques can be used based on sensitivity analysis procedures where few cases exist.

A second critical point is that the conclusion must be precise for accurate and useful comparisons. A conclusion drawn by the system must be classified as either correct or incorrect. Judgement substitutes statistical analysis. Weiss & Kulikowski (1984) as referenced by Klein & Methlie (1995) describe the process of performance validation as iterations through the following steps:

- Obtain the performance of rules on a set of stored cases
- Analyse the rules, and
- Revise the rules

The first step is to produce a performance summary for all stored cases. Several cases may give the same conclusions. Measuring the number of cases where the model's conclusion match with that of the expert, gives a performance measure called the true positive. The model may also infer a conclusion falsely. This performance measure is called the false positive. If all true positives were identified and

some extra false positives exist, it calls for strengthening of the antecedent conditions of one or more rules of the inference chains leading to the conclusion. This is called specialisation and makes the rule's conditions more difficult to satisfy and therefore to execute or fire. If no conclusions were identified as false positive and some positive conclusions were not identified, it calls for generalisation or weakening of some of the antecedent conditions of one or more rules. The rule's conditions must be weakened so that it can be fired in more of the test cases. Statistics about each specific rule can be gathered and the performance analysed. On viewing these results, a revision of the rules called rules refinement can be carried out.

Errors of commission can be corrected by specialisation, that is adding constraints to the antecedents of the rules and so limiting the situations in which it will be fired. Rules that should have fired could be generalised making the antecedent conditions less restrictive to fire more easily. The above is an example of rule refinement. By applying generalisation, the number of correctly concluded cases may be increased, but this may result in increasing the number of misconcluded cases. Specialisation may reduce the number of misconcluded cases, but have the potential to reduce the target ratio.

7.3 Testing and evaluation of the KB-DSS

When a prototype is running, design specifications are tested, revised and new specifications added to accomplish needs that were not initially known. Development moves through a series of cycles before the system is finally ready for operation. This is a spiral operation having a cyclic pattern. Testing and evaluation are well-known tasks in model building and computer programming. Data, knowledge and models need to be validated and the quality of the system assessed. In testing and evaluating the system, norms are needed to test the actual behaviour of the system. Three kinds of norms exist:

- Theoretical norms: norms described by the normative decision theory
- Empirical norms: norms such as those prescribed by the expert model for problem solving, and
- Subjective norms: qualitative assessments of the system performance, either by users or experts

Theoretical and empirical norms validate the system and subjective norms evaluate the system. Verification and validation of knowledge-based systems are closely related to maintenance and learning. The typical building process of the knowledge-based systems is incremental. New knowledge is added to existing knowledge and may affect the existing knowledge. Inside the knowledge base, three kinds of knowledge can be changed (TEIRESIAS system by Davis & Lenat 1982, as referenced by Klein & Methlie 1995):

- Inference rules can be added to, modified or deleted from the knowledge base
- Concepts (objects, attributes and values) can be deleted, added or changed, and
- Control strategies can be changed

Klein & Methlie (1995) deal with the verification and validation of knowledge-based reasoning systems; discussing techniques to verify the logical correctness of knowledge bases including problems of completeness and consistency; discussing what and how to validate; and evaluating performance. These issues were discussed in Paragraph 7.2.2: p130.

7.3.1 A framework for testing and evaluation

This framework includes DSS features and ES features under the paradigm of decision support (Klein & Methlie 1995). ES and DSS differ in several aspects. Testing and evaluating these two systems are two very different tasks. DSS is difficult to evaluate, because of the diversity of DSS. A DSS is a set of resources, data and models that are placed at the disposal of the decision-maker. The individual decision-making's behaviour determines the usage of the system. Various decision-makers may use a DSS in different ways. There is no single criterion by which a KB-DSS can be evaluated. In ES, the predictive ability can be a measure of good or bad.

Table 7-2 A framework for KB-DSS (Klein & Methlie 1995)

Validation

1 Data input
2 Knowledge base: concepts and relations (e.g. rules)
3 Reasoning: strategies
4 Results: conclusions

System performance

1 Efficiency
2 Data entry
3 Output format
4 Hardware
5 Usage: how much, when
6 Man-machine interface

Task performance aspects

1 Decision-making: time, alternatives, analysis and participants
2 Decision quality
3 User perception: trust, satisfaction, usefulness and understanding

Business opportunities

1 Costs: system development, operating and maintenance
2 Benefits: income and reduced costs
3 Values: service, competitive advantage and training

Evolutionary aspects

1 Knowledge maintenance
2 Response time to change demands
3 Functionality of the development tool

The ultimate objective of DSS is to improve the effectiveness of decision-making. The effectiveness of decision-making is still unproven using measurements. Various DSS evaluation techniques are discussed in Paragraph 7.1: p126. A KD-DSS can be verified and validated, by testing and evaluating its knowledge-base parts.

Table 7-2 (p129) shows a summary of the test and evaluation framework for a KB-DSS as proposed by Klein & Methlie (1995). Knowledge base verification as well as knowledge base validation was discussed in Paragraph 7.2.2 (p130). Both are essential when validating the KB-DSS.

7.3.2 Performance evaluation

In this section, measurements are determined to evaluate the functionality of the KB-DSS. Evaluation requires an external reference to which the system can be compared. Because of the evolutionary nature of DSS, the ill structuredness of the DSS tasks, and the individual decision making behaviour, it is difficult to establish such a reference. No one criterion is likely to be sufficient to evaluate a KB-DSS. Klein & Methlie (1995) present categories of aspects of criteria to evaluate a KB-DSS:

- System performance
- Task performance aspects
- Business opportunities, and
- Evolutionary aspects

♦ System performance

Measures such as response time, availability, reliability, data entry, dialog, usage time, users and quality of the system support (documentation, training and more) can be obtained by observations, event logging and attitude surveys as discussed in Paragraph 7.1 (p126).

♦ Task performance aspects

These aspects are concerned with the functionality of the computer program in relation to the decision task. A distinction needs to be made between decisions and outcomes. A good decision does not necessarily lead to a good outcome. The quality of the decision depends on the correctness of the actions taken by the decision-maker. Task performance is measured by the quality of the decision more than the quality of the outcome. It is measured in terms of time spent to make the decision, alternatives measured and information searched. Some qualitative measures such as trust, satisfaction and understanding can be done.

♦ Business opportunities

The amount of resources used for a decision in terms of cost, and the effects of commitment in terms of benefit such as income and reduced costs, can be measured. Methods that can be used to measure business opportunities are cost/benefit analysis and value analysis (See Paragraph 7.1: p126).

♦ Evolutionary aspects

These measurements are more qualitative than quantitative and judgement is primarily done by evaluating aspects such as the system's ability to change, the functionality of the development tool and the availability of a system's builder.

7.4 Evaluating the prototype

In Paragraph 7.1 (p126), examples of measures of DSS evaluation were described. These measures are listed again in Table 7-3. It was mentioned before that the evaluation of the DSS should be built into the development process (See Paragraph 7.1: p126). It was also mentioned that the choice of the evaluation method will depend on the DSS and the impacts investigated (See Paragraph 7.1.9: p129). Recall that Sprague & Carlson (1982) divide the possible evaluation measurements into four categories:

- Productivity measures to evaluate impact of the DSS on decisions
- Process measures to evaluate the impact of the DSS on decision-making
- Perception measures the evaluate the impact of the DSS on decision-makers, and
- Product measures to evaluate the technical merit of the DSS

Since the advice-system is a prototype, and not in production yet, it would not be possible to evaluate the DSS' impact on the decision-makers and students. After implementation, a questionnaire could be given to decision-makers as well as the experts that usually assisted them before the DSS to determine the impact and success of the DSS.

Table 7-3Examples of measures for DSS Evaluation (Sprague & Carlson 1982)

Productivity measures
1. Time to reach a decision
2. Cost of making a decision
3. Result of the decision
4. Cost of implementing the decision
Process Measures
1. Number of alternatives examined
2. Number of analysis done
3. Number of participants in the decision-making
4. Time horizon of the decision
5. Amount of data used
6. Time spent in each phase of the decision-making
7. Time lines of the decision
Perception measures
1. Control of the decision-making process
2. Usefulness of the DSS
3. Ease of use
4. Understanding of the problem
5. Ease of "selling" the decision
6. Conviction that the decision is correct.
Product measures
1. Response time
2. Availability
3. Mean time to failure
4. Development costs
5. Operating costs
6. Maintenance costs
7. Education costs
8. Data acquisition costs

Measures such as the number of alternatives examined, the number of analysis done, the number of participants, the length in time the decision took and the amount of data can be automatically calculated from the application and the statistics displayed on the web page presenting the user interface. This can be used as ongoing evaluation to determine the success and relevance of the project. The explanations of the recommendations should convict the user that the decision is correct. This would support the perception measures. To prove the effect and usefulness the user will have to complete a questionnaire.

ILOG JRules reports the number of rules fired and the time the rules took to fire can be calculated. These values can be retained and statistics compiled. When in production, these measures could be an indication of a problem in the system and should be constantly evaluated. The decision recommendations of the KB-DSS and the end-result of the decision made by the decision-maker using the KB-DSS can be recorded and re-evaluated by the experts from time to time to verify the rules and the process.

A product measure such as response time could also be captured automatically. Response time depends on access to the database, the network, the rule set and the time the rules took to fire. Access to the database is done once, at the start of the student session, and then used by the rules in the knowledge base when necessary.

The KB-DSS could be cost-evaluated by determining all the relevant costs involved to produce and maintain a KB-DSS on the Intranet of the university. The main decisive factor in measuring the success of the DSS would be the decreasing number of queries from students that need assistance in selecting the best courses for the current semester. By keeping statistics, it can be proved that the system was worthwhile. Another way of evaluating this KB-DSS is to consider the criteria set in Paragraph 4.1 (p41). These criteria were:

- The DSS must include an inference engine in their knowledge component to provide reasons for actions taken for example it must include an expert subsystem with an inference engine
- The knowledge component must include all relevant knowledge in a manner that course experts can maintain it
- Changes to the knowledge component should preferably not result in a new version of the DSS/SDSS software application
- Possibility to deploy the software application on the university's Intranet will be an advantage, and
- The DSS must be able to provide a reason for actions taken

Table 4-2 (p52) listed these criteria as met by ILOG JRules. The criteria re-evaluated are:

- ILOG JRules includes an inference engine in its knowledge component. It is implemented by using the Rete algorithm. JRules is an expert shell that provides an empty knowledge base and an inference engine. It offers the capability of a user interface that may contain the other components of a DSS or KB-DSS.

- In Paragraph 4.5.5 (p71), the rule editors were evaluated. It was concluded that the rule administrator would need some initial assistance in compiling the rules. He would then be able by selection, copying, deleting and pasting, to maintain and test rules after he received some training. The training should focus on the verification of the rules. When a new type of rule is needed, it would be best if the rule analyser consult the builder/programmer for assistance.
- Each department would have its own rules in a separate rule set file (.irl). Depending on the degree the student is pursuing the correct .irl file would be invoked. When updated and replaced by the department the next student enquiry when getting his data from the database will invoke the updated file immediately. The maintenance of the rules, except for structural changes to the objects used (addition of new fields etc.), should be fairly painless provided that the rule base was tested sufficiently by the rule administrator before submitting it into production. Setting and testing of the rule bases are done in a decentralised way. Each department functions independently of the others in setting and testing their rule base. When satisfied, their .irl file is exported and used with the application in production e.g. the Intranet where the students would have easy access to the application.
- The DSS application can run as an applet on the university's Intranet, and
- Reasons for recommendations, approvals and rejections are given as part of the user interface

7.5 Summary

Although the rule administrator will not be able to create his complete rule base using the current prototype, he can do rule maintenance and be responsible for his current set of rules. The administrator will be part of the testing and setting of the rules and the more experienced he becomes the less involved in the process the builder/programmer would be. The administrator will take the responsibility of the rules in his rule base, as set by him or with the assistance of the builder/programmer, or suffer the consequences when having to assist students when the system behaves unexpectedly. When the rules in the KB-DSS are verified as complete and consistent, the experts in the departments would be alleviated of the multitude of queries and allowed to focus on those situations where their expertise are really needed.