The application of IKBS technology to software design measurement and improvement

H. Sharp\textsuperscript{a}, G. Robinson\textsuperscript{b}

\textsuperscript{a}The Open University, Walton Hall, Milton Keynes MK7 6AA, UK
\textsuperscript{b}British Telecom Research Laboratories, Martlesham Heath, Ipswich IP5 7RE, UK

ABSTRACT

This report investigates current applications of Intelligent Knowledge-based Systems (IKBS) technology to the problem of software design measurement and improvement, and provides a comparative evaluation of six different approaches to this area, based on a common framework of technical and business criteria. Although there are few applications of IKBS in this area, those which do exist illustrate the different directions which future research could follow, and are encouraging enough for such paths to be explored.

INTRODUCTION

The importance of producing a quality design for a software system has been recognised for many years, and ideas of what a ‘quality design’ is have matured considerably since the notions of information hiding and structured programming were first introduced. One approach to improving the quality of designs is to provide quality measures which can be used to pinpoint weaknesses and deficiencies and therefore to judge how ‘good’ a design is. Unfortunately, qualitative, heuristic approaches are more prevalent than quantitative or algorithmic ones; even if numerical measures of design quality (metrics) are used, their ultimate interpretation and translation into an improved design is best left to a human expert.

Such a problem area appears to be an ideal one for IKBS (Intelligent Knowledge-Based Systems) treatment [1, 2]. However, few attempts have been made so far to produce knowledge-based support for software design measurement and improvement although it is clearly an area which would benefit from such support.

This paper investigates the current applications of IKBS technology in this area and provides a comparative evaluation of these applications based on a common framework of technical and business criteria. Assessing the suitability of particular measurement techniques or metrics has not been an aim of this paper. Also, the wider question concerning our current understanding of design quality is not discussed.
here. Instead, the authors have concentrated on the application of particular IKBS techniques within the design measurement and improvement field.

Six representative applications, with varying approaches to support are discussed and compared. Following the discussion and evaluation of current applications, the future direction of research activity in this area is discussed and specific suggestions for both long and short term research are made. Finally, a number of conclusions are presented in the last section.

Conventional approaches to software design measurement and improvement

‘Design measurement’ is still in its infancy, most software measurement tools being restricted to measuring code rather than design representations. Also, a key difficulty in introducing metrics into the mainstream of the software industry is the time and effort required to analyse the metrics data.

‘Design improvement’ has become a target area for many CASE tool manufacturers, but it is usually offered in the form of better tools for applying existing design methods, rather than as a set of techniques for optimising existing designs. Re-engineering tools, such as Software Refinery from Reasoning Systems Inc.[3], provide a means of re-generating a design from existing code, but no help is provided for improving the design itself; the main practical contribution to design improvements often consists of providing retrospective documentation.

Simulation based design improvement tools Tools for software (and hardware) design that incorporate or are based upon simulators are widely available. The simulation facilities offered might be said to correspond to ‘design testing tools for debugging’ although they are more commonly entitled ‘design validators’. They allow the user to examine how well a future system (or part of it) is likely to perform, by simulating the system’s behaviour and outputs when given specific inputs.

Generally, such tools also include static analysis capabilities for consistency and syntax checking, according to the particular design representation methods which are used. Examples of such tools include Athena systems ‘Foresight’, TRI’s ADAS, and AI Ltd.’s STEM. Petri net analysis is the principal technique used to simulate performance, and in one approach, ‘ESPNET’, an expert system using rules representing the Petri net model of the design is utilised to carry out the simulation. Verilog’s GEODE toolset which also falls in this category incorporates a code generator as well, and thus transcends the category of mere design improvement so as to encompass system generation.

Design metrics Design metrics have tended to relate to the process architecture of software designs, and usually measure the degree of coupling or cohesion that occurs in systems. Such measurements are generally interpreted as indicating aspects of a system’s maintainability; either the ease with which changes can be made, or the extent to which changes are likely to ripple through a system are the target characteristics of such measures. The exceptions to this trend are those
size metrics which can be applied to designs, such as DeMarco’s Bang metric [4] or the more generally known Function Point measure [5].

One factor that has limited the take up of such metrics has undoubtedly been their restriction to code measurement, since by the time that the metrics become available (after the code has been produced), it is too expensive to act upon any interpretation that may be placed upon them. In any case there are few commercially available tools that provide such metrics even at the code analysis stage.

Some currently available CASE tools incorporate design measurement to enhance or manipulate user generated designs internally, but their use is fairly simple. Automatically generating metrics from a given design is offered by some tools, e.g. MCASE from Mentor Graphics, but the user is left to interpret their significance.

**Analysis techniques for metrics data** There is no magical path leading from metrics data to better designs; metrics data has to be analysed before it can provide useful information. Applying conventional statistical analysis to software metrics requires specialist knowledge and is very time consuming. To get the most out of such work, a statistical analysis tool is required and even then, interpretation of the results is crucial to successful application of the knowledge.

**CURRENT APPLICATIONS OF IKBS TO SOFTWARE DESIGN MEASUREMENT AND IMPROVEMENT**

This section considers current applications of IKBS technology to the problem of design measurement and improvement. As already mentioned, there are very few reported applications in this area, and those which do exist either provide simple support, or are research projects which have not yet reached commercial maturity. Their objectives are therefore different, and it is difficult to compare them, but it is useful to have some form of comparative evaluation of their existing status and potential in order to identify fruitful avenues for further work.

**The Applications**

Six representative applications are presented, each of which offers a different approach to design measurement and improvement support, indicating the diverse nature of efforts in this area.

**Automatic design measurement and improvement: Knowledge-based Design Assistant (KDA)** KDA is a prototype system which was developed to explore the feasibility of applying IKBS techniques to software design evaluation and refinement, based on the Yourdon and DeMarco approach to software design [6]. The system is built around the blackboard architecture [7], with each ‘expert’ programmed to evaluate the design according to one design measurement technique and to suggest refinements which will improve the design.

The system operates as follows. The user enters an initial design for the system, and weights the evaluation criteria to indicate their relative importance. Once this initial design has been input, KDA repeatedly evaluates and attempts to refine it until the system is unable to perform any further refinements, or until a certain level of quality is
reached. This quality level is determined by a scoring system which uses the set of weights supplied by the user to produce an overall score of the 'goodness' of the design.

The evaluation and refinement portion of the system is designed to interface with a set of CASE tools for entering the design information. Although the measurement techniques currently included in the system are primitive, the architecture on which it is based can accommodate any number and type of measurements which are applicable to this style of design notation, subject to speed of execution and other performance considerations.

Quality advisor: MASCOT Design Support Environment (MDSE) MDSE was produced as a result of Alvey project SE/044 which was concerned with building an experimental design support environment for use with the MASCOT development technique [8]. The collaborators for this project were Ferranti Computer Systems Ltd., Logsys Advanced Systems, University of Stirling and YARD Ltd.

The subsystem of interest in this context is the Advisor subsystem, developed at the University of Stirling, which was intended to assess the static quality of a MASCOT design, supported by other elements of the tool [9]. As suggested by its name, this subsystem supports the designer by advising on which areas of the design are potentially problematic. The result of the project was a prototype tool in need of further refinement, including enhancement of the information presented to the designer.

The information required to assess the design is extracted directly from the diagrams and other design documentation which is entered using another section of the tool set. A simple rule-based knowledge representation expresses three sets of rules which are used to assess the design and produce a textual output for the designer. These are MASCOT rules, which are concerned with MASCOT syntax and semantics, metrics rules, which are based on coupling, information flow and control complexity metrics [10], and heuristic rules which were gleaned from design experiments conducted with groups of designers.

Design critiquing and argumentation: Design-oriented domain environments (DODE) This kind of system offers a designer two paths towards developing a design: design by construction and design by modification. Therefore, the environment contains design building blocks and a library of complete existing designs. As a designer is constructing an artifact, the system criticises the design to highlight problem areas. The critiquing interface is very powerful, since it is able to argue about the criticisms made using domain knowledge which the designer can easily understand, to substantiate its claims.

A number of these environments have been built for areas of design outside software as well as within it: interface design, graphics programming, voice dialog design, computer network design, kitchen design [11]. In the comparative discussion below, we have not chosen to work with just one of these environments, but to consider the basic concepts of the approach.
Extracting, interpreting and learning: the BT system At British Telecom (BT) a prototype design metrics tool which is capable of measuring data flow diagrams, entity relationship diagrams, and structure charts showing system architectures, has been produced and used experimentally as a component of a design system. It can measure electronic design representations produced automatically by a commercial CASE tool. It allows the user to collect a range of size and information flow metrics for the purposes of assessing designs according to a range of pre-determined criteria, or based on threshold values for individual measures.

Currently, these assessments are made available to the user within a ‘designer’s assistant’ environment which incorporates a rule based advice system, using rules representing heuristics for producing ‘good’ designs. These rules were derived from a knowledge elicitation exercise conducted with experienced designers within the company.

Work is progressing to link the advice given to the user directly to the design measurements by applying advanced data analysis techniques to the design metrics data. Neural nets, data mining techniques and data visualisation approaches are being harnessed so as to incorporate both a learning and an explanatory facility into the measurement component. These techniques which have already been shown to be effective when applied to code metrics data, should allow the design support tool to provide the user with reliable design improvement recommendations that have been generated automatically from design measurements.

Metrics interpretation: GALILEO GALILEO started life as ESPRIT project ESP/300 [12], and grew into a commercially-available system. It is available from BNR Europe and is an integrated set of tools to support the use of metrics in software development; this includes initially specifying and defining product quality, planning the use of metrics, monitoring software development and assessing the final product. The capability for detecting and diagnosing problem components, which is part of the monitoring function, contains an expert system. The problem components are detected statistically, and the data is fed into the expert system which provides an interpretation of that data. This interpretation would include the reasons why the component is a problem, and a recommendation for removing the problem; the system may ask questions of the user in order to provide this interpretation.

The knowledge contained in this expert system is encoded in simple production rules, and the reasoning strategy uses a system of weighting in order to reach its conclusions; a limited amount of unsophisticated fuzzy logic is employed within its reasoning strategy.

The system has been evaluated favourably by a limited number of people and is still being refined.

Syntax checking: Visible Analyst Workbench This system is marketed by Visible Systems Corporation of America. It provides a CASE tool set for working with the Yourdon and DeMarco or Gane and Sarson development notations. One of the modules contained in this set is an expert system for monitoring the development of the software design,
ensuring consistency in naming conventions, consistency of data flows through diagram hierarchies and proper symbol usage.

This expert system component is not sophisticated and does not incorporate any form of statistical or other metrics-based analyses. The knowledge which it contains is concerned with the syntax of the design notations, and therefore the support it provides is limited; it does not include any sophisticated ideas of the quality of the design itself.

THE COMMON FRAMEWORK

The common framework for discussing the six applications introduced above consists of a list of questions which address various interesting or desirable system characteristics. The list is restricted to questions for which positive, as opposed to vague or unhelpful, answers could be given for the systems being considered. This means that some interesting questions have not been asked.

There are two distinct perspectives which make up the framework itself: the technical view and the business view. The former looks at the level of sophistication of the application, and its technical abilities; the latter considers the likely impact which using the system would have at the business level of the organization. Questions relating to specific business benefits such as how quickly the costs involved in introducing the application will be recouped are difficult to answer for the systems which are still undergoing research or development, since the answers would depend on how the ideas are actually applied, and the business strategy of the evaluating organization. On the other hand, since design measurement and improvement is an important activity within software development, any real help which a system can provide will automatically improve customer satisfaction. Since the product, i.e. the software system, will be improved, it could be argued that any useful system will provide substantial business benefits.

The list of questions and the related discussion is presented below, followed by a table containing a summary of the discussion, the last two rows of which give a general assessment of the long and short term benefits which can be gleaned by using the system. The reader is advised to interpret this table with caution, because of the diverse nature of the systems, and the consequent difficulty in applying the criteria consistently.

Technical criteria

- Is the system a research prototype (P), under active development (D), or a complete, working system (C)? KDA, the BT system and DODE are all prototype systems. The DODE ideas have undergone quite extensive testing, but are not available commercially. Extending MDSE into a commercial tool is being considered, while GALILEO and Visible Analyst are both available commercially at present.

- Is the user interface to the system of high quality, i.e. easy to understand and use? There is no evidence that particular attention has been paid to evaluating the usability of the interface of any of the systems, although since MDSE and Visible Analyst are accessed through CASE tools, it is reasonable to assume that some usability evaluation has been
done. By the developers' own admission, the interface for MDSE needs improvement. The interfaces for DODE and the BT system are both good.

- **Is the knowledge representation used by the system sophisticated or simple?** All of the systems use simple rule-based knowledge representations.

- **Does the system glean the data it requires from documentation (D), i.e. automatically, or the user of the system (U), i.e. interactively?** MDSE and GALILEO exploit both documentation and users for relevant information at appropriate times. The BT system requires guidance from its user, but its measurement data is extracted automatically from the design representation entered through its associated CASE tool; KDA also glean information from the internal representation of the design, but is guided by factors entered by the user. DODE bases its critiquing advice on internal design information, but it processes information from the user in supporting its explanation and argumentation functions.

- **What type of decision support does the system provide?** This question was highlighted in the earlier section containing system descriptions. KDA can be used to generate alternative designs from an initial specification and to refine them automatically. MDSE offers advice on possible problem areas of the design. The BT system only provides advice and an information capture service to the user. The expert system component of GALILEO interprets the results of statistical analyses of software components; it is targeted at the management side of software development. DODE provides design critiquing and argumentation support.

- **Can other measurement or analysis tools be interfaced with the application?** Both KDA and MDSE allow for the addition of further measurement techniques to their existing frameworks, the BT system can interface with other tools, and GALILEO can be tailored to handle different quality standards. It is unclear whether a DODE could be easily interfaced with a statistical analysis system; the advantages of doing so are also debatable. Since Visible Analyst provides only simple evaluations which are based on the design notation being used, it is not possible to interface its output easily with analysis tools, and it is unlikely to be worthwhile to try and integrate such capabilities into the system.

- **Does the system incorporate a learning facility?** Sophisticated learning cannot be incorporated into any of the systems easily, since it is likely to require a change of knowledge representation. However, the BT system is designed to incorporate a metrics data-based learning capacity; neural networks have provided the most promising results so far in this sphere.

- **Would it be possible to analyse statistically the output from the system?** This evaluation depends on whether numeric measures of design quality, i.e. design metrics, are used in the system, and if so which ones. Visible Analyst and DODE do not use such measures, KDA can be extended to include such measures, MDSE does use such measures, and the BT system and GALILEO are both specifically designed to process statistical data.
**Software Quality Management**

- **What is the level of technological innovation?** One innovative angle taken by KDA and MDSE is the encoding and automatic use of design heuristics. KDA was one of the first systems to use the blackboard model of control in the area of conventional software design support. Similarly with GALILEO and the BT system, it is the kind of knowledge which has been captured rather than the techniques used which is innovative. Visible Analyst’s technology and knowledge is much less innovative. The DODE approach to design reuse and argumentation is very novel.

**Business criteria**

- **Does the system deliver real, quantifiable benefits?** As mentioned above, any system which provides true support for the design measurement and improvement activity will automatically be beneficial to the end product, i.e. the piece of software being developed. DODEs have been developed successfully for a number of areas within software design; because of the powerful domain links, these environments offer real benefits. However, the limiting factor is the need for domain knowledge. KDA, MDSE and the BT system require further development before real benefits can be guaranteed; all of these systems, however, embody powerful ideas which will provide real benefits if developed. GALILEO can provide real benefits for software development by identifying at the design stage program segments which will cause problems during the maintenance phase of development. The benefits of using Visible Analyst depend partly on the experience of the user of the system. If someone new to the design notation is using Visible Analyst, then the system will contribute more to the design process than if someone with years of experience of the design notation is using it.

- **How does the system perform when subjected to cost-benefit analysis?** As already suggested, this kind of analysis is difficult to perform for those systems currently undergoing further research and development. However, certain criteria which impact on this question can be considered:

  - **How large is the problem being tackled?** KDA, MDSE, DODE and the BT system address fairly substantial problems. GALILEO is more specialised, and Visible Analyst tackles a comparatively small problem.

  - **How much benefit will be provided?** DODEs provide a powerful design environment incorporating reuse and criticism, two key elements of effective software design. The work connected with MDSE and KDA will provide insights into the design process, although both need to be developed further. It is fairly clear from earlier discussions that the benefit provided by Visible Analyst is limited and dependent on the system’s users and their experience. GALILEO provides the most immediate benefit of the six systems described here since they contain substantial and scarce metrics knowledge. The use of GALILEO, which is commercially available, enables the developer to improve software quality and the chances of avoiding maintenance problems; it also utilises knowledge which may not be available to the developer from any other source. The BT system also contains substantial metrics knowledge.
## Table 1. A summary evaluation

<table>
<thead>
<tr>
<th>Criterion</th>
<th>KDA</th>
<th>MDSE</th>
<th>DODE</th>
<th>The BT system</th>
<th>GALILEO</th>
<th>Visible Analyst</th>
</tr>
</thead>
<tbody>
<tr>
<td>System status</td>
<td>P</td>
<td>P</td>
<td>D</td>
<td>P/D</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>User interface</td>
<td>Poor</td>
<td>Needs improving</td>
<td>Good</td>
<td>Quite good</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Knowledge representation</td>
<td>Rules</td>
<td>Rules</td>
<td>Rules</td>
<td>Rules</td>
<td>Rules</td>
<td>Rules</td>
</tr>
<tr>
<td>Knowledge source</td>
<td>D/U</td>
<td>D/U</td>
<td>D/U</td>
<td>D/U</td>
<td>D/U</td>
<td>D</td>
</tr>
<tr>
<td>Support type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interface to other tools</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Learning capacity</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Statistical analysis</td>
<td>Possible</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Business benefits: long term</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Business benefits: short term</td>
<td>Low</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>
How soon will these systems have a significant impact on software development? Any environment which is based on a collection of knowledge becomes more effective as the knowledge base is expanded. In this sense, IKBS systems will not have a significant impact in their early years. Clearly, GALILEO and Visible Analyst which are available commercially can be used now. New prototype DODEs are being developed quite rapidly, but the domain knowledge bases must have time and relevant exposure to develop. The impact from the research associated with the BT system and with KDA depends on the speed with which the use of design measures and improvement strategies are understood in more depth, and the amount of resources available for development; an appreciable impact could be felt within the next few years. Similarly, the impact which MDSE could make is dependent to a degree on the amount of resource expended towards research into the design process. However, if the system were to be refined to commercial standards as it is, then for a project currently using MASCOT, the use of MDSE would be noticeable almost immediately.

FUTURE DIRECTIONS

When considering the application of IKBS techniques to the area of design measurement and improvement, there are two different questions to be tackled. These are, what kind of knowledge should be encoded in the system? and what kind of support should the system offer? Obviously, these two are closely connected, but in trying to identify worthwhile future research directions, both of these perspectives should be considered. Before these are discussed, however, it is worthwhile looking at one other question: ‘Is there a need for knowledge-based systems in this area?’.

Does the need exist?

Before getting too involved with a discussion of the direction in which research and development in this area should go, it is worthwhile considering whether the journey is necessary or not. The first observation to make is that software development is still in need of improvement, therefore attempts to develop quality software more effectively are necessary. Improving the understanding and measurement of design quality is an important goal relating to this issue, as has been acknowledged for a long time. The second observation is that, subject to its suitability, it is perfectly legitimate to apply any available technology to bring about such improvement. This, coupled with the fact that the area of software design in general and design measurement and improvement in particular bear the characteristics of topics suitable for IKBS treatment, makes it clear that applying IKBS techniques to design measurement and improvement is appropriate and desirable. This is not the same as saying that it is necessary, however, and the suitability of other techniques should also be considered. For example, some would argue that the transformational approach to software development [13] renders this argument redundant.

Statistical analysis of design properties has produced some successes; the work associated with GALILEO, among others, has
proved this to be so. Heuristic, qualitative ‘measures’ such as the techniques of structured programming and information hiding which can be used to produce quality software have also proved to be effective. The problem with qualitative measures, however, is that it is difficult to communicate and explain their use to those who do not have any practical experience of using them.

In answer to this question, therefore, the following points should be considered:

1. The efficiency and quality of software development needs to be improved;

2. All available, suitable technologies should be brought to bear on the problem of improving software development efficiency and quality;

3. Software design measurement and improvement is a topic suitable for IKBS treatment, and therefore should be considered in this light.

What kind of knowledge?

Many different kinds of knowledge are used by a designer while designing a software system; we have a long way to go before we understand the process to any degree of certainty. The list includes relatively simple knowledge, such as the syntax rules of a particular design notation, more advanced, but general design knowledge such as ideas about coupling or complexity, and more specialised knowledge such as heuristics pertaining to the application of these general design guidelines in specific application domains.

Considering the systems described earlier in this report, it can be seen that Visible Analyst incorporates relatively simple, syntax-based knowledge, GALILEO contains more sophisticated knowledge about metrics interpretation, MDSE has a collection of knowledge types including design heuristics pertinent to a specific application domain and syntax-based knowledge, KDA incorporates general design measurement and refinement knowledge, and DODE bases its power on broader domain knowledge and existing designs.

Ultimately, the goal of research should be to identify and encode as much of all of these as possible, however Rome was not built in a day, and the process must begin somewhere. Suggestions for short term research are given below.

What kind of support?

Expert systems which rely on rule-based collections of knowledge and forward or backward chaining reasoning have become quite well-accepted in commercial circles. As with every new technology, there has been a period of trial in which users demanded evidence of the technology's durability and applicability, but for basic expert systems, that period is over. In this kind of system, the rules encoded in its knowledge base must be the root of its power since the reasoning technique is so simple. However, there are more sophisticated architectures, knowledge representations and reasoning strategies available. If such techniques are used instead, then more sophisticated
support which utilises deep knowledge and deep reasoning from the design domain can be provided.

Again, considering the systems described earlier in this report it can be seen that Visible Analyst and GALILEO provide only standard expert system support while DODE, MDSE, KDA and the BT system use more advanced techniques as well; GALILEO provides more sophisticated support than Visible Analyst because of the depth its knowledge.

**Short term research**
The best direction for short term research, providing returns and usable systems quickly is to concentrate on providing standard expert system support for restricted problems within design measurement and improvement. The emphasis in this approach lies in the identification and encoding of domain knowledge, which will provide insights into the deep knowledge and deep reasoning behind design; these insights can then be fed into more ambitious, longer term research projects.

The measurement and improvement problem can be restricted in a number of ways, some of which were exemplified by the systems described earlier. For example, by concentrating on specific design notations (e.g. Visible Analyst), by limiting the field of vision to particular design metrics (e.g. GALILEO), by identifying measurement and improvement heuristics for particular application domains (e.g. MDSE), by limiting the scope of design activity being considered (e.g. data design, database design, user interface design etc.).

The main hurdle to be overcome is finding the knowledge source for the problem identified; in practice, it is more likely that the knowledge available will dictate the problem rather than vice versa. Problems associated with knowledge acquisition have been documented elsewhere [2], however when dealing with design measurement and improvement there is an extra difficulty which was summed up by a GALILEO development team member who said that “very few people understand about metrics”; it would be fair to add that very few people understand about design either.

**Long term research**
In considering longer term research, there are many avenues which open up. The first point to make is that results of the short term efforts may indicate particularly fruitful directions to be pursued. However, long term research should concentrate on experimentation with more advanced IKBS architectures and/or knowledge representations but again within restricted problems; tackling the whole question of design measurement and improvement is too wide to be solved realistically without substantial groundwork.

DODEs will continue to have a great impact on research in this area. MDSE and KDA have shown how the use of existing, documented design practices combined with results of experimental studies can give rise to sophisticated support based on a fairly simple knowledge representation. Insights gained by these projects can now be fed into
more advanced studies to expand their findings, increase the collection of heuristic knowledge, and hence produce more sophisticated support.

Extra dimensions which could be added to this kind of experimentation include software reuse, software prototyping and automatic learning, for example using neural networks. The main hurdle to automatic knowledge acquisition, or learning, in the past has been the inability to measure accurately properties relevant to design quality. Advances in this area have now been made, and with the advances in neural network technology, this approach holds greater promise.

At the KBSE'92 conference, the most important hurdle to acceptance of knowledge-based software engineering systems was identified as being the application to real problems. Until it can be shown that new technologies are effective within the real world of software development, they are unlikely to blossom. This was seen with regard to CASE technology, and it is happening again with IKBS technology. One necessary element to any long term research project, therefore, is to address large, real world problems.

CONCLUSIONS

Using IKBS techniques to produce sophisticated support for the design measurement and improvement activity is an important avenue of research which should be pursued. Although there are few current applications, those that do exist illustrate the different directions which future research could follow, and are encouraging enough for such paths to be explored.

The reasons for the limited number of systems include a lack of understanding regarding software development in general and the design process in particular, but also the immaturity of the more sophisticated IKBS techniques themselves. Through a policy of short term research aimed at clarifying design knowledge, and long term research aimed at exploring both the design process and IKBS technologies, progress towards further support environments can be made.

REFERENCES


790 Software Quality Management


