SOFTWARE ENGINEERING/IKBS
Strategy for Knowledge Based IPSE Development

INDUSTRY

MOD

DTI

SERC

The Alvey Directorate

Prepared by
Tony Dignan (Software Engineering Directorate)

Alvey Directorate
Dept of Trade and Industry
Millbank Tower
Millbank
London SW1P 4QU

August 1984

SOFTWARE ENGINEERING/IKBS
Strategy for Knowledge Based IPSE Development

1. INTRODUCTION

The Alvey Software engineering Strategy (1) identifies three generations of Integrated Project Support Environment (IPSE). Those three generations are characterised as follows:

1st generation — file based
2nd generation — database
3rd generation — knowledge based.

A target date of 1989 is proposed for the design of a demonstration third generation IPSE. This document is concerned with the development of "intelligent" tools for use within this third generation IPSE or Information Systems Factory (ISF).

Within the context of this programme intelligent tools may be loosely defined as those which utilise techniques which are normally classified under the general heading of IKBS. Such tools may replace conventional tools (e.g. a rule based syntax checker), enhance conventional tools (e.g. intelligent front end to a complex test generator) or provide assistance for tasks which have little, if any, conventional tool support (e.g. expert system advisor for requirements analysis).

2. OBJECTIVES

The basic objective of the intelligent tools programme is to ensure that the UK has the expertise needed to meet the 1989 target date for the design of a knowledge based IPSE. In order to achieve this objective three interdependent sub-goals can be identified:

(1) The industrialisation of IKBS techniques—the development of IKBS products must be advanced from a largely experimental process to one which is predictable and controllable.
(2) The convergence of Software Engineering and certain IKBS themes—products must be developed which incorporate elements of SE and IKBS and the system developer must be familiar with a wide range of techniques which originated in either community.
(3) The development of tools which exploit IKBS techniques in order to support systems development.

The third of these sub-goals, the development of intelligent tools, provides the most tangible product. For this reason the intelligent tools programme will be the main vehicle for achieving the overall objective.

A secondary, although vital, objective of the programme is to ensure that the benefits of intelligent tools are sufficiently widely appreciated that the knowledge based IPSE will find an eager market. The programme will seek to disseminate information on tools as they become available and encourage the widespread adoption of such tools.

3. TOWARDS A DIALOGUE

Although there are several different software development life-cycle models in existence there is general agreement upon the basic structure of such a model. It consists of a number of, largely sequential, phases covering requirements specification, design, implementation, integration and on into operation and subsequent "maintenance". While compliance with this basic life-cycle model does not guarantee a successful project it has been found that non-compliance, in order to shorten delivery timescales or save costs, is likely to result in a spectacular failure. The life-cycle model provides an essential framework for project management, it enhances the visibility of the project and allows progress to be monitored. Quality management and configuration management activities are also largely based upon the life-cycle model.

Much of the current Software Engineering research work is aimed at improving the working practices within each phase of the life-cycle and in particular at improving the end-product from each phase. The use of more formal methods and notations ensure that the end products from each phase can be checked for internal homogeneity and for compatibility with the output from the previous phase. Such methods, which promise to provide worthwhile improvements in software quality and productivity, will be progressed within the Alvey Software Engineering programme.

In the ideal life-cycle model the work within each phase is complete, and the end product "frozen", before the subsequent phase begins. In practice, certainly for any large complex system, such an ideal is rarely, if ever, attained. Even if it were possible to define fully the requirements prior to design the user would continue to refine his perception and expectations of the system during the development process. Clearly such changes to the requirements must be incorporated into the delivered system if it is to meet the user's real needs. Despite various feed-back loops on the basic life-cycle model it must be admitted that current software engineering practice is less than totally successful at coping with such continuously evolving requirements.

IKBS offers a somewhat different paradigm for system development. One which places great emphasis on the production of animated specifications or prototypes. Such specifications enable the user to explore the consequences of his stated requirements and are claimed to be easier to modify than conventional software specifications. Such a paradigm appears to avoid some of the problems inherent in the present Software Engineering approach. However, the IKBS paradigm is immature and unproven. It may not deliver as much as it promises when exposed to the commercial and technical constraints of the market place. Indeed it has yet to address, or even recognise, some of the legitimate concerns of the software engineer.

Clearly the largely pragmatic, Software Engineering community and the largely academic, IKBS community have much to offer each other. At present there is comparatively little dialogue between the two communities. There is a need to introduce IKBS ideas and techniques to the commercial and industrial Software Engineering community and to demonstrate that such techniques can be applied to large scale systems. Equally there is a need to introduce the IKBS community to the Software Engineering approaches to such diverse topics as predictability, reliability, maintainability, etc. The intelligent tools programme provides a natural forum for the inter-community dialogue. IKBS techniques will be exploited, within normal Software Engineering discipline, to produce commercial quality artifacts. Such artifacts will then be available for use by both communities.

4. STRATEGY

4.1 Early Exposure

The introduction of new tools often leads to changes, in methods of working, which are difficult to predict in advance. Such changes in working methods may in turn modify the requirements for the tool. Hence there may be several iterations of build/use/modify before the interaction between a tool and its environment stabilises. During this period the tool is unlikely to provide the full anticipated benefits and may be abandoned completely by its user community. It is likely that intelligent tools will have fundamental and often unpredictable effects upon the system development process. In order to assess the impact of intelligent tools and their likely influence on the methods and skills required for system development it is essential to obtain early experience on the use of such tools and feed the evaluation back into the tool development programme and other areas of the Software Engineering Strategy.

While 1989 is considered to be a realistic target date for the design of a prototype, knowledge based, Integrated Project Support Environment, it will be possible to introduce some intelligent tools before this date, in particular through the first and second generation IPSE programmes. Indeed, as explained above, it is essential, to the speedy exploitation of the knowledge based IPSE, that prior experience of intelligent tools be obtained. Similarly 1989 cannot be regarded as marking the end of intelligent tool development, it is likely that many difficult areas of system development will still be poorly served by tools at that date.

The intelligent tools strategy will, therefore, like the overall Software Engineering strategy, encourage a continuous programme of research and development coupled with the exploitation of emerging tools at the earliest possible opportunity. However, within this continuum, three distinct phases can be identified. Each of these phases is discussed below.

4.2 Exploratory Phase

Some intelligent tools are already being developed and the programme will encourage the production and early use of these early tools.

It is natural that many of these early tools will tackle some of the less demanding areas within system development and project control, areas which are unlikely to have the highest potential payoff. Similarly the tools are likely to come from a variety of sources in an unco-ordinated manner and cannot be expected to form a coherent or well matched set. Consequently care must be taken to ensure that expectations, among the user community, are not raised too high as this could damage the whole tools programme.

Within the constraints discussed above, the programme will encourage the introduction of intelligent tools, in a controlled manner, to assist with various tasks within the first generation of Alvey IPSE (it should be noted that the first generation of IPSE are intended to have an evolutionary capability such that more advanced tools can be added). The use of these tools will be carefully monitored and, in conjunction with the software metrics programme, attempts will be made to measure their effectiveness. A wide spectrum of software development organisations will be encouraged to use these intelligent tools and the results (good and bad) will be widely disseminated.

4.3 Method Based Phase

As indicated in the Alvey Software Engineering Strategy the use of an Integrated Project Support Environment implies the selection, by the user project, of a particular development methodology. With a knowledge based IPSE it is vital that the methodology and IPSE are compatible since the in-built rules will assume a certain approach to system development; any significantly different approach will, at best, be poorly supported by the IPSE and may prove to be impossible.

While it would be possible to construct some form of knowledge based IPSE in the absence of a clear model of the underlying methodology (or classes of methodology) to be supported the value of such an exercise is questionable. A prerequisite for the implementation of a successful third generation IPSE is seen to be a clear definition of the methodology to be supported, with the environment being constructed to assist, encourage and enforce adherence to the methodology. This is not intended to imply that an IPSE should totally constrain its users to a particular mode of working; some degree of freedom must be provided to allow a project team to adapt the IPSE to cope with local conditions and individual skills, etc. The environment must also be capable of evolving to accommodate improved techniques and tools as these become available; a characteristic which the knowledge based IPSE must share with the earlier generations. Nevertheless the basic approach should be first the method then the toolset.

Unfortunately, there are, at present, no methodologies which have gained wide acceptance. Indeed even within a single organisation several different methods and sets of tools may be used. This situation is likely to impede the development of a commercially viable knowledge based IPSE. This second, method based, phase of the intelligent tools development strategy is hence closely linked with the progress made by the formal methods programme (2). There will be considerable interaction between the two programmes to identify a common set of objectives which are both desirable and achievable.

Even without an explicit intelligent tools initiative various IKBS techniques would naturally be absorbed by the software engineering community as the impetus towards formal methods gathers pace. For example, work on declarative languages will be exploited in the formal specification of system requirements; work on theorem proving will provide for designs which can be proven correct with respect to the specification; work on inference will be exploited to provide analysis and prototyping facilities. The intelligent tools programme will encourage and seek to accelerate such synthesis. However the programme will also seek to extend the impact of IKBS into areas of Software Engineering where the natural take-up of such techniques might be less rapid. For example intelligent front ends can be constructed for a wide range of tools addressing all aspects of system development and project control while expert systems can provide assistance with many of the creative tasks which currently have no support.

While formal methods can be seen as a sound foundation for the use of IKBS techniques the widespread use of more formal methods is likely to be considerably assisted by the success of the intelligent tools programme. Tools will be required to enable the large numbers of practising software developers, trained in traditional methods, to adopt the more rigorous approaches. Thus there is a natural synergy between the intelligent tools and formal methods programmes.

4.4 Convergence Phase

The convergence of the enabling technologies, at a number of different levels, is essential to the ultimate success of the Alvey initiative. Throughout the 5 years of the Alvey programme there is likely to be considerable and increasing cooperation between the various communities together with the adoption, by one community, of ideas and techniques pioneered by another. The Integrated Project Support Environment, intended to support all phases and activities of project development, will be the most visible product of this convergence. However, given the acute problems each community faces in attempting to develop its own technology, it is inevitable that the full convergence will only become a reality towards the end of the period.

The convergence of Software Engineering and certain IKBS themes is a major goal of the intelligent tools programme. This convergence is not confined to the use of intelligent tools to support traditional Software Engineering tasks, this is merely the major vehicle for encouraging the convergence process. Within industry it should be possible to merge elements of both communities to produce a new generation of software developer able to utilise a wide range of tools and techniques which originated in either Software Engineering or IKBS.

The Alvey/SERC Infrastructure, by supporting a common base of hardware and software for use by both communities, encourages a dialogue to take place. The intelligent tools programme will implicitly assist with the convergence process by encouraging members of both communities to be contributors to and users of the tool set. More explicitly the programme will seek joint ventures, with the IKBS arm of the Directorate, which seeks to exploit IKBS research within the context and constraints of industrial system and software development.

The programme will also seek to identify and produce tools to support MMI and VLSI design activities within the IPSE. Efforts will also be made to define standard interfaces which will assist with the communication of ideas and with technology transfer between the various enabling technologies.

5. APPLICATION AREAS

The Software Engineering Strategy identifies (section 3.3.4) six main subsystems which together will form an ISF, namely:

(1) Specification and prototyping facilities.
(2) A Software Development Environment.
(3) A facility for CAD of VLSI and hardware development.
(4) A database or knowledge base of available software and hardware components.
(5) The communication systems, both local and wide area, to facilitate cooperative development.
(6) Project management aids.

Intelligent tools can make a major contribution in all six of these areas.

5.1 Specification and Prototyping

The software industry has, to some extent, accepted the argument that greater effort devoted to the early phases of the development life-cycle will result in a better quality product and an ultimate saving in through-life costs. Consequently there is considerable interest in more rigorous approaches to requirements analysis and specification and the use of formal, machine manipulable representations which can be subjected to analysis to prove internal consistency and completeness. That such interest has not been translated into the widespread adoption of more formal methods is due to several reasons, not least of which is the difficulty of learning and applying such methods. This problem is perhaps most severe for large complex projects involving large, mixed ability teams, i.e. the very projects which stand most in need.

The use of models or prototypes to verify design assumptions and provide user feedback is well established in many branches of engineering. Prototyping has often been proposed for software based developments, indeed some methodologies place considerable stress on the desirability of early prototyping. However, in practice large scale prototypes are rarely, if ever, constructed, largely due to the cost of such an exercise and the anticipated detrimental effect upon the timescales for delivery of the final product.

Clearly prototyping should result in a product which is more closely matched to the user needs; only by interfacing with a model of a proposed system can a user assess how it will impact upon his environment and thus what the detailed requirements are likely to be. The increased emphasis on MMI aspects of systems development also implies an enhanced need for prototyping; it is clearly difficult to claim that a system is user-friendly if no users have been allowed to evaluate the system.

Logic programming and functional programming are approaches, with roots in IKBS, which appear likely to revolutionise the software life-cycle by supporting the development of animated specifications, which in some cases may be sufficient in themselves to remove the need for programme implementation. The full potential and limits of these approaches need to be fully explored.

5.2 Software Development Environment

The software development environment is currently the most advanced subsystem of the ISF and the one where intelligent tools are likely to make the earliest impact. This scenario is to be encouraged since it would ensure that the tool builders could be among the first users.

Initially individual tools will be introduced as they become available and will provide only fragmented, ad hoc support. Later environments will contain intelligent tools which provide integrated support for the complete development life-cycle.

For many applications, in particular within the smaller systems market, intelligent programme generators are likely to be developed which will replace both the present generation of generators and many standard applications packages. Such generators will be easier to use than existing products, cover a wider range of applications and provide more options.

The impact on larger systems is also likely to be dramatic. All aspects of software development are candidates for intelligent tool support, however the most significant developments are likely to affect those tasks which currently have little or no tool support. For example the derivation of a design to satisfy a requirement specification is currently without tool support. Expert systems advisors to assist with this task are feasible even if total automation is unlikely.

Intelligent tools for Software Development will contribute directly to one or both of the Software Engineering Strategy goals of improved quality and improved productivity.

5.3 CAD for VLSI

The Alvey Directorate strategy for VLSI (3) includes a number of initiatives in the area of Computer Aided Design (CAD). Initially this work will be largely independent of the work pursued in the Software Engineering area although a measure of compatibility will be provided by adherence to the Infrastructure Policy and Common Base Policy as appropriate; this compatibility should be sufficient to ensure that any early CAD tools can be integrated into the IPSE programme as this progresses.

Intelligent tools for CAD is a topic of considerable interest and activity and more active coordination of the CAD and Software Engineering programmes is required to reduce duplication of effort and to ensure a smooth integration.

Tools for CAD are likely not only to contribute to improved quality and productivity but to reduce the time taken to produce a chip or path. At this stage of requirements specification VLSI and software have identical needs and a common approach would appear feasible. Other areas, such as chip layout and path analysis, are unique to CAD but can be supported within the IPSE development.

5.4 Database/Knowledge Base of Components

The use of formal specifications of Behaviour will assist the wider re-use of software based components. However other information also must be recorded (the environment it requires, timing/performance information, how easily it can be modified to perform slightly different tasks). Having established what to record and how, it is then necessary to encourage users to record available modules and to look for suitable re-usable modules, intelligent tools are likely to assist with both activities.

There is little incentive for the developer of a software based module to enter details into the knowledge base hence this task must be as simple as possible and an intelligent front end to the recording tools could assist.

For the potential user of an existing module the incentive to search the knowledge base is obviously greater. However, software developers often display a great aversion to using the fruits of someone else's endeavours; in the early days the hit rate will not be high and any difficulty in interrogating the knowledge base could destroy the scheme. Hence intelligent browsers to enable the user to easily scan a library and identify potentially useful modules are essential to the success of the approach.

5.5 Communications Systems

While local and wide area networks open up many opportunities they also provide many potential problems; intelligent tools can ameliorate some of these problems.

For the user with a large volume of incoming mail an intelligent filter to prioritise the messages and possibly to dump irrelevant messages could transfer the communications system from a time consuming monster into a useful tool.

For the system supplier, intelligent route finders and fault finders would greatly ease maintenance problems and could increase the capacity of a system.

5.6 Project Management Aids

Intelligent tools for project management may include some for the very earliest phases of the life-cycle. Current algorithmic approaches to estimating, require fairly precise specification of the system to be developed, something which is only available after considerable work has been done. A rule based approach is likely to provide a useful, if less accurate, estimate much earlier in the process.

6. PLAN

This section describes the activities which must be performed in order to achieve the three phases of delivery identified in section 4.

6.1 Exploratory Phase

The Alvey Directorate will organise joint Software Engineering/IKBS workshops and other events to accelerate the two-way technology transfer between the SE and IKBS communities.

Useful, knowledge based, tools which can be developed using current technology will be identified. Some tools will be commissioned for use within the first generation of IPSE.

Jointly, with the Director for IKBS, on-going research work which will affect the intelligent tools programme will be identified and monitored.

6.2 Method Based Phase

Develop tools, identified by the formal methods programme, to support mature methods.

Identify areas of Software Engineering most in need of new or improved tools (e.g. from the STARTS guide (4)) and how IKBS can contribute.

Promote work which will explore the potential and limitations of the IKBS development paradigm within the context of commercial and industrial system development.

Jointly, with the Alvey Director for IKBS, identify problem areas within IKBS product development where proven Software Engineering techniques may enhance the quality of the product or the productivity of the development team and commission work to prove the efficiency of such techniques in an IKBS context.

6.3 Convergence Phase

Initiate research into the likely life-cycle model for future SE/IKBS products and the implication for methods and tools.

Initiate research on the development of large computer based systems to identify the knowledge which will need to be incorporated into the third generation IPSE.

REFERENCES

(1) Talbot D., Witty R.W.
Alvey Programme—Software Engineering Strategy
DTI Publication, November 1983.

(2) Jackson M.I., et al.
Programme for Formal Methods in System Development
5th March 1984.

(3) Fawcett W.
Alvey Programme—VLSI Strategy
DTI Publication, December 1983.

(4) Various
The STARTS Guide
DTI Publication.