

## CHAPTER 10 CONCLUSIONS

### 10.1 Research Goals

It was the proposition of this research that there are a number of weaknesses in the current approaches being taken in the provision of software project planning tools and that there is significant scope to improve on existing systems in areas such as:

- **Creation of project plans** - few tools offer automatic guidance on the creation of technical and management plans. Of use to the project planner would be the automatic creation of an outline plan from specified (pre-defined) types of projects, which could be further refined to the particular project under consideration. In addition, users have to input large amounts of project plan data with little or no support for the accuracy, completeness and quality of the plan.
- **Decision support** - Most systems fall short of supporting the project planner in the decision making process. They do not offer assistance in representing knowledge about plans, or provide mechanisms for reasoning about plans in flexible ways.
- **Flexible Knowledge Base** - In the continuously evolving domain of software technologies, of great importance in the provision of automated decision support is the ability to dynamically update the knowledge base to cope with new and evolving standards and best practices.
- **Scenarios** - Although some tools offer 'what if' analysis in response to changing parameters, they do not offer direct 'recommendations' for action given a certain situation. Of use to the project planner would be the ability to create several different scenarios (or work breakdown structures) of possible future plan, based on the variation of a number of key project parameters, with associated decision support and plan verification. This would allow the project planner to examine alternate project plans from different perspectives.

- **Distributed cross-platform systems** - A further aspect to supporting the software project planner which is not addressed by today's support systems is the distributed and cross-platform nature of systems development. Users of existing software project planning systems could benefit greatly from support for the distributed cross-platform nature of modern client-server development.

This research set out to explore the role of artificial intelligence techniques in the provision of an intelligent assistant based software project planning tool which would address the issues described above. In addressing these aims, this research devised a framework and architecture which was used as the basis for the design of an intelligent assistant system. From this design a prototype system was implemented for use by software project managers in the planning of a distributed multi-platform software development project.

The main outcome of this was an architecture for an intelligent assistant system for use in software project planning. A prototype system was constructed to test the proposed architecture, and feedback from a series of end user trials by commercial tool users was evaluated to assess the usefulness and suitability of the system.

## **10.2 Research Outcomes**

This thesis reviewed the main approaches to providing intelligent assistance and proposed that in the complex domain of software project planning, a useful tool to support the project manager would be a hybrid of a number of techniques - Decision Support System, Expert System, Blackboard and Intelligent Agents - encapsulated within an agent-orientated framework. This approach enables the inter-working of a variety of well understood techniques within a single underlying framework.

This research also reviewed the typical services currently provided by software project planning tools and conducted a survey of tool users to highlight user need for an intelligent assistant within a software project planning tool and to further identify

possible shortcomings in the services provided by existing tools. The results of this review and survey were an increased understanding of the needs of software project planners and the perceived deficiencies in existing tools. This survey also provided supporting evidence for the usefulness of an intelligent assistant based system.

This research conducted a review of a number of diverse architectures based around the agent-orientated paradigm and proposed a set of desirable architectural characteristics - which should be taken into consideration when developing an architecture for an intelligent assistant system - and compared the proposed architecture against these characteristics, demonstrating that the system satisfied these criteria.

Following this investigation, Java and CORBA were adopted as suitable solution technologies to implement the architectural properties of distributed platform-independent systems. Furthermore, JESS was put forward as a suitable agent language for the purpose of implementing a prototype system within a Java-CORBA architecture.

In order to validate this research, a prototype application based on the proposed architecture was developed. The successful construction of this application demonstrated that the architecture was implementable and could be deployed in a commercial setting. Furthermore, this facilitated the end user testing of the application which concluded that the system provided a novel approach to the creation of project plans and had potential to assist and support project planners with making project plans and the associated decision making.

### **10.3 Further Research**

This section will briefly outline some of the research directions which could be further pursued as a result of the research reported in this thesis. This list is not intended to be exhaustive.

### 10.3.1 System Architecture

The agent-based architecture described in this research provides a novel, open, flexible and adaptable approach to the implementation of an intelligent assistant system for in software project planning. However, there are a number of areas which could be further investigated.

The User Interface is a light-weight system component which handles the management of all the screen elements (menus, dialog boxes, etc.), validates data entered by the user and passes on clear functional messages to the rest of the system. The User Interface component is an ideal candidate for incorporation into a web browser, as the existing functionality could be re-implemented using Java applets. This would provide for a thin network client executing in a web browser with the remainder of the system located elsewhere on the network, thus extending the system to any client platform for which there exists a web browser.

The Agent Controller is a supervisory unit over the agent community which manages the scheduling and execution of agents. It contains the inference engine and related modules for the JESS system. This approach of allowing a separate inference engine for each agent language allows for a number of distinct inference mechanisms to be used, in which case they would reside as separate sub-components of the Agent Controller. A natural extension to this architecture would be to extend the number of inference engines, possibly to consider some of the other languages and knowledge representation schemes.

Agents are located in the Agent Library, but remain under the control of the Agent Controller, whose purpose is to manage the physical agents themselves and to service requests for agent interactions. This architecture allows for the updating of the knowledge base by adding, updating and deleting agents in the Agent Library. There are a number of strategies which may be employed to update the Agent Library, with the most promising being via the Naming and Trader service of CORBA. The CORBA Trader Service provides an 'advertising directory' for CORBA objects,

which allows objects to publicise their services and bid for jobs, whereas the Naming service provides a 'name directory' for objects, which allows applications to look up objects by name. Using these two CORBA services it would be possible to maintain an Internet site (single or multiple) which contains new agents and has a situation where the Agent Controller could interrogate the CORBA ORB to identify new (remote) agents. These agents could then be acquired and inserted into the local Agent Library and possibly be subject to an electronic commerce style of payment for such new agent.

### **10.3.2 Knowledge Base**

The knowledge base for this intelligent assistant system is encapsulated in a series of intelligent agents which are located in the Agent Library. In the previous section, a number of proposals for further work have been put forward. In addition there are also other knowledge-orientated aspects which warrant further investigation.

For the purpose of this research it was considered that the particular method of knowledge acquisition employed was not of primary concern. Rather, the method should elicit a sufficient quantity of useful data which could be used to construct agents. This research has used a combination of printed sources and informal interviews as the primary method of knowledge acquisition. At this point it is worth considering the appropriateness of these methods of knowledge acquisition within the context of this research. It would be useful to explore other methods of knowledge acquisition as well as the broader consideration of the knowledge lifecycle. For example, given a large set of agents (and thus elicited knowledge), some form of knowledge management procedure would be necessary. Another aspect of the knowledge lifecycle which would benefit from further consideration would be the validation of elicited knowledge prior to its incorporation into agents.

In relation to the knowledge base, the main issue which arose during the user trials was the request for advice which was more quantitative in nature. During the development of agents as part of this research, the request proved difficult primarily

because little suitable source material was available which contained quantitative data / results which could be used as the basis for agents. In addition, a barrier to developing such material would be the difficulty which humans have in discerning the differences between quantitative values at a fine grain level for less specific domains such as software development. For example, there is no appreciable difference between the values of 70% and 75% if they were expressed as a measure of suitability for a given lifecycle model. However, this request is an indicator of the nature of advice users perceive to be useful in addition to the advice already produced by the system. It is therefore worth investigating the issues surrounding the provision of such quantitative advice in conjunction with the previous issue of knowledge acquisition and validation.

The prototype system developed as part of this research contained a relatively small number of agents which operated in a small number of potential advice areas. To provide a more complete knowledge base a larger set of agents, covering a larger number of areas, would be necessary. For example, a series of agents dedicated to standards such as ISO 15504, CMM, etc., would be useful for users operating within those standards or for those considering adopting such standards.

The task of constructing agents (as described in this thesis) is a manual procedure and consists of first writing the agent header and subsequently constructing the agent rules in a suitable language - in this case JESS. Of potential interest would be automated tool support to assist agent developers (possibly user-organisations) in constructing their own agents. For example, the RISKMAN2 project (cf. section 4.2.3) developed an associated product, "Daemon Writers Pack" [Power, 94], to assist experts in risk management write Daemon's (RISKMAN agents). It consisted of a series of forms with a strict set of heading / formatting guidelines which acted as a generic rule template from which a developer would code a C++ Daemon. A major limitation of this approach was the complex implementation issues surrounding the translation of the text based forms into C++ Daemons.

To date all agents developed for the prototype system have been manually coded following the process described in section 7.3.6. The translation from a decision table

to JESS rule scripts is a reasonably straightforward process, as JESS IF-statements follow a clear and simple format. It should therefore be possible to create a 'user friendly' GUI based tool which allows a user to enter conditions (dependent token values) and associated actions (advice to be generated), and produce a JESS rule script in the appropriate agent format. As part of the P3, project an experimental prototype of a 'Agent Developer Kit' was developed using Visual Basic, which was capable of taking simplistic conditions / actions and producing basic JESS scripts. However, this was not pursued further for commercial reasons.

It should be possible to develop a series of add-on tools for end users with features for automatically generating agents via a GUI based system and the editing of existing agents. Such a system could also implement Agent Library housekeeping and other procedures. Such a tool would be of benefit to users interested in creating a set of agents dedicated to internal company standards or practices.

### **10.3.3 Prototype System**

For the purpose of this research, issues such as the execution speed of the system were not of primary concern, although a number of measures were taken to address such issues. However, a number of issues remain, mostly in relation to CORBA ORB initialisation, the launching of server objects and the presence of the OrbixWeb debug window. Successfully addressing these issues, would result in a subsequent prototype system operating at more acceptable levels (from a commercial perspective) and thus be in line with users comments received during the trials.

The addition of extended functionality within the prototype system, in particular enhanced scenario analysis and the provision of a larger set of agents would lead to a pre-commercial prototype system which could be used as the subject of further user trials to assess the commercial viability of an intelligent assistant system for software project planning.

During the users trials there was the suggestion of repositioning the system for use as a training tool, in which users could develop a model of a fictitious project and thus practice project planning skills on a 'virtual project'. There are a number of possibilities for such a repositioning which are worthy of further consideration, which include the possibility for an interactive (possibly web based) version of the P3 Handbook and Training Guide with a slimmed down version of the system acting in a project simulator / advisor role.

This research set out to examine the provision of an intelligent assistant system within the domain of software project planning. However, it should also be possible to extend this approach to the parent domain of software project management and the associated tasks of managing and tracking a software development project.

## **10.4 Concluding Remarks**

Due to the growing complexity of software systems and the current economic context, software projects are facing more and more constraining production objectives in terms of time, cost, quality and risk. This evolution in the nature of projects being undertaken by software organisations results in increased difficulties associated with planning, managing and executing software development projects. Indeed, software projects often fail because the project managers lack knowledge of good practices and effective processes which can reduce risk and increase the likelihood of success.

Existing project planning tools do give support to the project manager and have several basic strengths; planning calculation and re-calculation; recording progress and feedback data; comparison of planned against actual achievement and re-calculation of the plan in relation to progress update. These reflect the strengths of data processing by computers applied to project planning. However, such tools do not provide support for the project manager in the decision making process and do not offer assistance in representing domain knowledge about plans and designs or provide mechanisms for reasoning about plans and designs in flexible ways.

This research has proposed an agent-orientated framework as the basis for an intelligent assistant system for use by software project planners and designed a novel architecture upon which this system can be constructed. This architecture is based on a fusion of a number of techniques within a multi-agent framework which aims to improve the quality of the decision making process of software project planners. This framework incorporates the information gathering and analysis techniques of a Decision Support System with the ability of an Expert System to propose possible solutions using expert knowledge and best practices and the power of Blackboard to exchange information between components. This novel approach enables the interworking of a variety of well understood techniques within a single underlying framework - that of the agent-orientated paradigm.

To assist with validating the proposed architecture, a prototype application was developed and a series of trials conducted. The conclusion of these trials was that the prototype system demonstrated that the notion of an intelligent assistant system for software project planning was a viable concept, worthy of commercial investigation. Further, it demonstrated that the proposed multi-agent framework provided a viable architecture for supporting decision making which has the potential to be of use in a commercial setting.

This research is a significant step forward in the development of a new generation of software project planning tools. The approach described in this thesis is a fusion of a number of well understood techniques within a single unifying framework. An important characteristic of this approach is the combination of these techniques in an open distributed environment with the potential for continuous evolution.