

CHAPTER 1 INTRODUCTION

1.1 Background

This thesis describes a multi-agent based architecture for an intelligent assistant system for use in software project planning. The research explored the role of artificial intelligence techniques for automated tool support as applied to software project planning and in particular addressed the issues of knowledge capture and reuse in such a tool environment. This research also examined the issues surrounding the emerging requirements for support systems for distributed multi-platform software development projects. In addressing these aims, this research devised a framework and architecture for use as the basis for the design and construction of an intelligent assistant system and implemented a prototype system for use by software project managers in the planning of a distributed multi-platform software development project.

The following sections describe the motivation and background of the work, both from a user-level and system-level perspective, leading to a discussion of issues concerned with a new architecture for intelligent assistant systems. The statement of the aims and objectives of the research and an outline plan for the rest of this thesis complete the chapter.

1.2 Software Project Management

The Project Management Institute defines project management as [PMI, 96];

“The applications of knowledge, skills, tools and techniques to project activities in order to meet or exceed stakeholders needs and expectations from a project”.

Project management is an integrative endeavor - an action, or failure to take action, in one area will usually affect other areas. The interactions may be straightforward and

well understood, or they may be subtle and uncertain. For example, a scope change will almost always affect project costs, but it may not affect team morale or product quality. These interactions often require trade-offs among project objectives - performance in one area may be enhanced only by sacrificing performance in another. Successful project management requires actively managing these interactions.

Many techniques of general project management are applicable to software project management, but Brooks [Brooks, 87] pointed out that the processes and products of software projects have certain characteristics that make them different. One way of perceiving software project management is as the process of making visible that which is invisible [Hughes and Cotterall, 99]:

- **Invisibility** - when a physical artifact such as a road is being built the progress can actually be seen. With software, progress is not immediately visible.
- **Complexity** - Per dollar, pound or euro spent, software products contain more complexity than other engineering artifacts.
- **Flexibility** - The ease with which software can be changed is usually seen as one of its strengths. However this means that where the software system interfaces with a physical or organisational system, it is expected that, where necessary, the software will change to accommodate the other components rather than vice versa. This means the software systems are likely to be subject to a high degree of change.
- **Standard Process** - in other engineering disciplines with a long history the processes are tried and tested. Our understanding of software processes has developed significantly over the past few years, however we still cannot predict with complete certainty when a particular software process is likely to cause development problems.

Software project managers are responsible for planning and scheduling software development. They supervise the work to ensure that it is carried out to the required standards and monitor progress to check the development is on time and within budget.

1.3 Software Project Planning

Software project planning is an integral part of the software project management activity. Its objectives are to provide a framework that enables the project manager to make reasonable estimates of resources, costs, and schedule [Pressman, 97]. These estimates are made within a limited time frame at the beginning of a project and should be revised regularly as that project progresses. In addition, estimates should attempt to define 'best-case' and 'worst-case' scenarios so that project outcomes can be bounded.

Effective management of a software project requires thorough planning of its progress. The project manager must anticipate problems which may arise and prepare tentative solutions to those problems. A plan drawn up at the start of a project, should be used as a driver for the project. The initial plan is not static but must be modified as the project progresses and new information becomes available. Project planning is probably the activity that takes most management time [Sommerville, 95]. The planning process starts with an assessment of the constraints affecting the project. The progress milestones and deliverables are then defined and a schedule drawn up. Project managers revise their assumptions about the project as more information becomes available.

1.4 Difficulties Facing Software Project Managers

Due to the growing complexity of products and commercial systems, large projects are facing more constraining production objectives in terms of time, cost, quality and risk. This evolution in the nature of projects being undertaken by software organisations has resulted in increased difficulties associated with planning, managing and executing software development projects.

One of the key issues is decision making. Software project managers make many decisions every day, ranging from the relatively inconsequential to the significant.

Such decisions are based on a combination of judgement and information from staff, clients, research literature and current market forces, as well as knowledge gained from previous projects. Ideally, all relevant information should be brought together before judgement is exercised. The quality of a decision depends on the adequacy of the available information, the quality of the information, the number of options available at the time of the decision and the ability of the people involved to interpret this information.

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. Managers of projects need to know how to establish a set of processes which are tailored to a project's requirements in terms of time, cost, quality and their associated risks [Ould, 90]. A desired outcome of this research is a planning tool which will increase the likelihood of success by helping the project manager who has to make decisions on these issues. Such a tool will encapsulate expert knowledge and make it available to all users. Some of the potential benefits of this approach as applied to the decision-making process in the domain of software project planning are:

- Suggestions are made which help the user balance cost, quality and time in making decisions about the use of project resources.
- Knowledge is shared about different lifecycle models and why one or another may be more suitable for the users projects.
- Measurements are suggested which will enable the user to see how well the project is reaching greater organisational goals and re-plan the ways to reach these goals, if necessary.

Even the most experienced project manager may have difficulty knowing the best planning options, even if the critical input parameters of resources, constraints and requirements are known.

1.5 Intelligent Assistance for Software Project Planning

The notion of an intelligent assistant is not new. Indeed, as far back as 399 BC Socrates claimed to have an intelligent assistant, although not in the strictest sense of course. But Socrates did claim to have a non-human companion, which he called a Daemon. Intelligent and always ready to offer good advice, Socrates' daemon could be trusted to act without prompting. Real, hard-coded, linguistic and symbolic links abound between Socrates daemon and today's notion of an intelligent assistant [Leonard, 98].

[Boy, 91] offers the following characterisation of an intelligent assistant system:

“In an aircraft cockpit, a human copilot shares the work, but not the ultimate responsibility, with the captain. The captain is the master on board: he may consult the copilot at any stage but will take the ultimate decisions. If the captain delegates a part of his responsibility to the copilot, then the copilot will take this delegation as a task to be executed. In addition, the captain may at any time choose to stop the execution of a task by the copilot, if he judges it to be necessary. However, a copilot may have personal initiatives, for example, testing parameters, keeping current with the evolving situation, predicting deducible faults, etc. A copilot may process the knowledge included in the operation manual on his own initiative or at the request of the captain. He should be capable of explaining, in an appropriate amount of detail, the results of his processing.”

Weld [Weld, 95] suggests that a software system designed to act as a team member could help in the planning and execution of a project. Such an intelligent project assistant could help to preserve knowledge about tasks, to record the reasons for decisions and retrieve information relevant to new problems. They could function as co-workers, assisting and collaborating with the design or operations team for complex systems. They could also supply institutional memory. They could recall the rationale of previous decisions and, in times of crisis, explain the methods and reasoning previously used to handle that situation.

In software development projects, an intelligent project assistant could keep track of specifications, design proposals, and implementations for a software project throughout its life cycle. It can record the design decisions of a constantly changing team and also be a repository of solutions for new projects. Reasoning techniques can be used to track the (mis)match between specifications and implementations, while analogy techniques can be used to look for existing specifications, components or implementations that match some new requirement.

An intelligent project assistant can additionally be of benefit when training new personnel. For many tasks, on-the-job training is extremely effective, providing the trainee with the chance to make real, on-the-spot decisions and see the consequences. On-the-job training is impossible, however, when a bad decision can be disastrous - as in the planning of a large complex software development project. Simulations of the project planning process, would enable the development of training systems for such situations [Grosz and Davis, 94]. These same simulation capabilities are also important when the cost of assembling large groups of people for training is prohibitive.

1.6 Aims and Objectives

One aim of this research was to understand the complex decision making process associated with planning a software development project. Additionally, the development of an intelligent assistant system to assist project managers in their decision-making process was planned. A prototype system was constructed to test the proposed architecture and feedback from trials by commercial tool users evaluated to assess the usefulness of such a system.

This research started from the standpoint that there is significant scope to improve on existing software project planning systems and in particular to provide capabilities such as;

- The provision of advice to assist project managers in the decision making processes associated with the formulation of project plans.
- The ability to reason about a project's plans, analyse alternatives and select the most suitable course of action.
- The ability to assimilate knowledge and best practice.
- The ability to assist the project manager in adherence to standards, industry best practices and implementation of company policy.
- The ability to dynamically update the knowledge base.
- The ability to cope with new and evolving standards and best practices.
- The ability to manage and analyse large amounts of project data.

1.7 The P3 Project

The implementation and testing of the prototype system was conducted within the scope of the P3 (Project and Process Prompter) project [O'Connor et al., 97a].

The P3 project was funded by the fourth framework programme of the European Commission as ESPRIT project 22241 (cf. Appendix A). The two main deliverables of this project are a "Handbook and Training Guide" and a pre-commercial prototype decision support tool "Prompter".

The Handbook and Training Guide [Catalyst, 99] is designed as a standalone document which requires no other documents or tools to be useful and has two main components:

- **Volume 1** considers process planning as it relates to anyone starting a project, i.e. the basic processes that a project manager needs to know.

Part I contains a high-level view of some of the general challenges of project management. Part II takes the project manager from setting up to closing down the project. Parts III, IV, V, and VI are the technical details of project management, and Part VII examines the pros and cons of following some well-known standards.

- **Volume 2** includes models and decision processes to assist the project manager which were incorporated into the Prompter tool.

The prototype of the Prompter tool has implemented the decision models above to assist project managers in the planning of a software development project. Its aims are to provide project managers with a greater understanding of options available during planning and why one choice should be made over another. Prompter gives the project planner the opportunity to input project goals and certain project-specific variables, match them against a generic model to create a specific project model, then analyse a set of options which may be used to organise the project so that it will meet its goals.

This researcher's role in the P3 project was that of project manager in charge of the overall architectural design (as described in chapter 6) and implementation of the knowledge base (agents) for the Prompter tool [O'Connor and Renault, 98]. As, such this project provided an ideal framework within which to implement the proposed architecture, utilising the knowledge from the Handbook and Training Guide.

1.8 Layout of Thesis

In this chapter the motivation and objectives of the work have been explained. Chapter 2 describes the domain of software project planning in more detail in order to understand its unique characteristics and assess what special considerations are necessary when developing intelligent assistance systems. The findings of a survey of tool users are also presented and analysed in conjunction with a discussion on existing project support tools to assess the need for, and usefulness of, the integration of intelligent assistance in software project planning tools. Chapter 3 provides a review and discussion of approaches to supporting intelligent assistant systems, leading to a proposed architecture for applying intelligent assistance to the domain of software project planning. Chapter 4 presents a critical review of intelligent assistant systems, from both a user and system architecture perspective. Chapter 5 discusses issues relating to the design and development of a knowledge base, including knowledge representation and acquisition. Chapter 6 details the proposed architecture for an

intelligent assistant system based on the system proposed in Chapter 2 and the issues discussed in subsequent chapters. Chapter 7 describes the design and construction of a prototype implementation of the system. Chapter 8 provides an overview of research methodologies and describes the approach taken to the validation process. Chapter 9 presents a strategy for trial usage by a group of commercial users and discusses the lessons learned from user feedback gained from these trials. Finally, Chapter 10 contains the conclusions and the recommendations which describe the advances made in this research.