A COMPUTERIZED TUTOR FOR ARCHITECTING SOFTWARE
Supporting the Creative Aspects of Software Development

José L. Fernández-Sánchez and Javier Carracedo Pais
Industrial Enginneering School, Madrid Technical University, José Gutierrez Abascal 2, 28006 Madrid, Spain

Keywords: Intelligent agents, software architecture.

Abstract: CASE tools must be more user-oriented, and support creative problem-solving aspects of software engineering as well as rigorous modelling based on standard notations such as UML. Knowledge based systems and particularly intelligent agents provide the technology to implement user-oriented CASE tools. Here we present an intelligent agent implemented as a CASE tool module. The agent guides the software architect through the architecting process, suggesting him the actions to be performed and the methodology rules that apply to the current problem context.

1 INTRODUCTION

Software engineering is an established discipline of engineering where standards, notations, processes and best practices are currently defined.

The software development process consists of a mixture of creative and mechanistic activities some of them performed by humans and the others well supported by tools.

The creative activities of the software architecting phase of a real-time system development are mainly those related to: the identification of system responses, the identification and selection of the architecture components, and the solving of the time and concurrency problems frequent in these applications.

Current CASE (Computer Aided Software Engineering) tools can guide software architects in the software architecting process at the supported methodology level, basically they can check consistency of software models and in some cases verify methodology rules, but they are rather inefficient at the development process level and software architect level.

CASE tools must be more user-oriented, and support creative problem-solving aspects of software architecting as well as rigorous modelling.

Knowledge based systems and particularly intelligent agents provide the technology to implement user-oriented CASE tools where an intelligent agent implemented as a tool module, guides the software architect through the architecting process, suggesting him the actions to be performed and the methodology rules that apply to the current problem context.

Based on our previous experience of developing PPOOA (Pipelines of Processes in Object Oriented Architectures) method and tool based on UML (Unified Modeling Language) notation for architecting real-time systems (Fernandez 2003), we try to demonstrate that constructing a computerized tutor for assisting to the development of the architecture of a real-time system is feasible.

We begin the paper describing the characteristics of intelligent agents and their role in software engineering activities. We then present PPOOA_ATA (Architecting Tutor Agent), its capabilities, the main components of its architecture and how it is used. We then close the paper with some conclusions and how we plan to extend the architecting tutor agent and the CASE tool with new features.

2 AGENTS AND SOFTWARE ENGINEERING

Authors involved in agent research have offered a variety of definitions for an intelligent agent. It is not the purpose of this paper to survey the definitions of intelligent agents but to identify what is the essence of an intelligent agent, which are its

L. Fernández-Sánchez J. and Carracedo Pais J. (2007).
A COMPUTERIZED TUTOR FOR ARCHITECTING SOFTWARE - Supporting the Creative Aspects of Software Development.
In *Proceedings of the Second International Conference on Software and Data Technologies - SE*, pages 367-370
DOI: 10.5220/0001340003670370

main properties and how an intelligent agent can be applied in the software engineering domain.

2.1 Intelligent Agents

The essence of an intelligent agent is formalized by Franklin, when he defines it as an entity situated within and a part of an environment, that monitors that environment and acts on it over time, in pursuit of its own plan and so as to change what it monitors in the future (Franklin 1996).

Franklin also identifies the properties that may have an agent:

- Reactive: responds in a timely fashion to changes in the environment;
- Autonomous: exercises control over its own actions;
- Goal-oriented: may act following a plan;
- Temporally continuous: is a continuous running process;
- Communicative: can establish dialogs with other agents including people;
- Adaptive: sensitive to each user´s strengths and weaknesses;
- Mobile: able to transport itself from one machine to another;
- Flexible: actions are not scripted;
- Character: apparent personality and emotional state.

The intelligent agent definition given above satisfies the first four properties. Adding other properties may produce useful types of agents for example, mobile learning agents.

2.2 Applying Agents to Engineer Software

Effective intelligent agents must deal with the grey areas of the incomplete function for which it is an approximation. Software engineering is one of these grey areas where the adequate output for a given input is context-dependent. That is, there are different solutions for the same software problem (requirements) and the suitability of a solution (design) depends on the context (project constraints).

The knowledge for building software is buried in books and manuals or in the heads of software engineering experts, and how to find and get access to it is a challenging task. Frequently the software engineer is blocked in one step of the development process, having no access to the human expert that can help and suggest what the software engineer needs to know at this particular situation.

The availability of a personal computerized tutor is not time restricted and can help the software engineer in three ways: by capturing a significant part of the developing process knowledge; by reasoning based on this knowledge and received events; and by automating the application of this knowledge to the software under development. Presenting relevant information and proposing suggestions eases the decisions made by the software engineer.

Figure 1 represents the interaction of the computerized tutor, the software engineer and the CASE tool in the scenario of generic software development process. The computerized tutor asks questions to the software engineer and receives responses from him. The computerized tutor checks the software models and receives events from the CASE tool, for example when a new building element is dragged and dropped in a diagram. Following the development goals and reacting to events, the computerized tutor sends suggestions to the software engineer. So, the expert is available when the software engineer has a need.

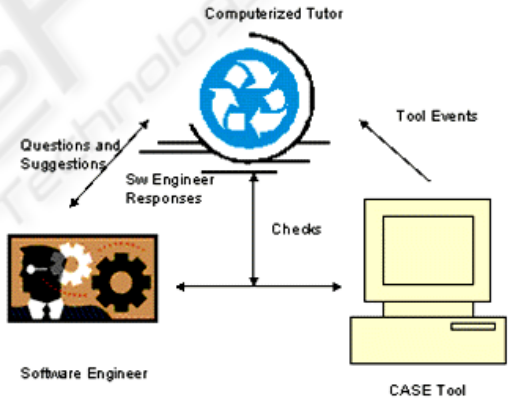

Figure 1: Computerized Tutor in Software Development.

The feasibility of the usage of computerized tutors is also supported by recent experiences, which have similarities and differences in goals, scope and implementation with respect to our computerized tutor.

Diaz Pace and Campolo report an expert system being constructed that will support design modifiability (Diaz Pace 2003). Bachmann, Bass, Klein and Shelton propose an expert system that collaborates with the architect to produce a design of an architecture that supports an expected change (Bachmann 2004). This rule based architecture design assistant uses architecture modifiability attribute models viewed as frames. The next

attribute they plan to add will be performance. WayPointer is a commercial tool that provides an integrated solution for implementation of best practices for requirements engineering, modelling, documentation and testing of software. In WayPointer, an intelligent agent is responsible for helping the software engineer to accomplish a well-defined task. The knowledge of the agents is captured by a set of rules that are defined in an application specific rule language (Racko 2004).

3 ARCHITECTING TUTOR AGENT

PPOOA_ATA (Architecting Tutor Agent) can be considered a hybrid agent exhibiting two different forms of reasoning: one reactive or event driven, and other proactive based on action planning and the architecting process execution for achieving the software architecting goals.

We describe below PPOOA_ATA, its capabilities, the main components of the agent architecture and how the agent interacts with the user.

3.1 PPOOA_ATA Capabilities

The current prototype of PPOOA_ATA supports the following capabilities:

- Has knowledge about the PPOOA architecting process and the architecting guidelines and is able to use this knowledge to give suggestions to the software architect;
- Can proactively take action based on the architecting goals and the architecting process implemented as a plan;
- Can reactively take action based on the interoperation with the PPOOA-Visio CASE tool;
- Can inform the software architect about the current step of the running architecting process or subprocess;
- Help issues regarding PPOOA are presented to the software architect depending on the current context;
- Provide configuration parameters to adapt the agent to a particular software architect;
- The software development project information is maintained independent.

3.2 Components of the PPOOA_ATA Agent

The known agent metamodels (Henderson-Sellers 2005) and the integration of the agent with the PPOOA-Visio tool are the main drivers of the architecture solution selected for the PPOOA_ATA Agent.

The agent is integrated in a Windows environment using the Microsoft Foundation Libraries to communicate with the software architect through windows and dialogs.

The agent is implemented as an add-on that communicates with PPOOA-Visio tool using the COM interface provided by the Office suite.

The agent is coded in Visual C++ and it executes in its own thread. The agent is autonomous but it can open a Visio instance.

The main components of the agent are represented in Figure 2 and described here:

- Agent controller managing the current situation based on events and planning next steps;
- Knowledge DB keeping the knowledge regarding the process and guidelines;
- Configuration DB keeping the configuration parameters adapted to the user;
- Project DB keeping the information of a particular software development project;
- Interface Visio Tool receiving events from PPOOA-Visio tool and communicating with it;
- Architect Interface is the man machine interface.

These components have dependencies among them as represented in the figure. The agent controller centralizes the control over the rest of the components and interoperates with the PPOOA-Visio CASE tool and the PPOOA help files.

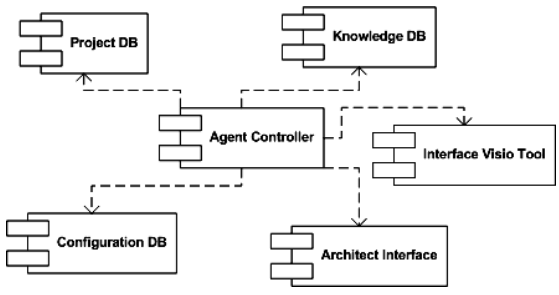

Figure 2: PPOOA_ATA Architecture.

3.3 The Agent in Use

The PPOOA_ATA agent presents a main screen, shown in Figure 3, which is part of the computerized tutor and provides outputs relative to the architecting process, next goals, help issues and methodology guidelines to apply. This main screen acts as a real-time library reference tuned to the exact step of the architecting process at which the software architect is working.

The main screen shown in Figure 3 describes the current step, goals and substep of the architecting process for a particular situation. In this situation the computerized tutor recommends the software architect to begin with step 3 of the PPOOA architecting process. The first substep of the step 3 is concerned with the identification of the real-time system external events and their arrival patterns (periodic, bounded, bursty, etc.). The buttons shown in Figure 3 give the software architect the opportunity to know the help issues and guidelines applicable to this situation. These screens are not presented here.

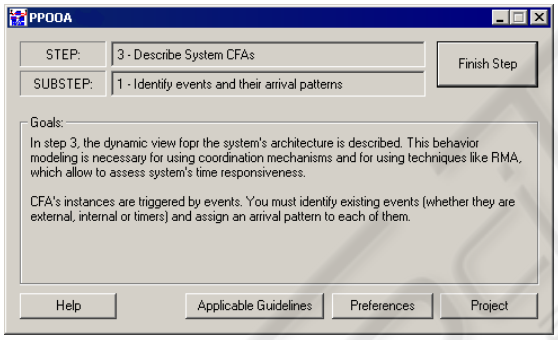

Figure 3: PPOOA_ATA Main screen.

4 CONCLUSIONS AND FUTURE WORK

Our main goal was transform PPOOA-Visio software architecting tool into a more user centered CASE environment. The implementation of an agent acting as a computerized tutor was one of the main achievements.

We tested the agent with some examples of architecture development and we believe that it performs well regarding tutoring novice software architects based on its interoperability with the PPOOA-Visio CASE tool. We plan to extend the agent and the CASE tool in the following directions:

- Extend the tool events received by the agent;
- Enhance model checking capabilities of the tool and allow the agent to interpret model checking information and take actions;
- The CASE tool should offer a wide range of real-time software architecture patterns for reuse.

REFERENCES

Bachmann, F., Bass, L., Klein, M., Shelton, C. 2004. Experience Using an Expert System to Assist an Architect in Designing for Modifiability. *Fourth Working IEEE/IFIP Conference on Software Architecture.* IEEE.

Diaz Pace, J.A., Campo, R.M. 2003. Design Boots: Towards a Planning-based Approach for the Exploration of Architecture Design Alternatives. *Argentine Symposium on Software Engineering.* SADIO.

Fernandez-Sanchez, J.L., Martínez-Charro, J.C. 2003. Implementing a Real-Time Architecting Method in a Commercial CASE Tool. *16th International Conference on Software and Systems Engineering and Their Applications.* CNAM.

Franklin, S., Graesser, A., 1996. Is it an Agent or just a Program? A Taxonomy for Autonomous Agents. *Third International Workshop on Agent Theories, Architectures and Languages*. Springer Verlag.

Henderson-Sellers, B., Quynh-Nhu, N.T., Debenham, J. 2005. An Etymological and Metamodel-Based Evaluation of the Terms Goals and Tasks in Agent-Oriented Methodologies. *Journal of Object Technology, vol 4 no 2, March-April 2005, pp 131-150*. JOT.

Racko, R. 2004. A Cool Tool Tool. *Software Development Magazine. May 2004*. CMP Media LLC.