Report on the Workshop on the State of the Art in Automated Software Engineering

Yves Ledru
LSR/IMAG
Université de Grenoble
FRANCE
Yves.Ledru@imag.fr

David F. Redmiles
Information and Computer Science
University of California, Irvine
Irvine, CA, USA
redmiles@ics.uci.edu

On June 28, 2002, the Institute for Software Research at U.C. Irvine hosted a workshop on the state of the art in automated software engineering. Its goal was to bring together ASE researchers for informal presentations of current work, with the hope of identifying emerging trends and challenges.

The workshop started with an invited talk by P. Dourish. His talk pointed out the futility of requirements. In a context of ever evolving requirements, it is mandatory for software to be very adaptable. Automated software engineering may provide a solution to this flexibility demand, especially if deployed on the user's site. Requirements was actually one the major topics discussed at the workshop with talks by R. Hall and S. Fickas who point out growing challenges of validating requirements and systems.

Another major topic is to master the complexity of software engineering tasks and artifacts. This complexity is reflected in the diversity of models and notations used to describe software, tools and knowledge. It also requires the development of efficient tool support and tool infrastructure. These topics were addressed by the talks of J. Grundy, D. Redmiles, S. Henninger, F. Shipman and Y. Ledru.

These position papers are available on line[1]. The rest of this report gives their abstracts.

Abstracts of the position papers

Recent Experiences with Code Generation and Task Automation Agents in Software Tools (J. Grundy, J. Hosking) As software grows in complexity, software processes become more flexible yet complex, and more developers must co-operate and co-ordinate their work, software tools providing developers editing, reviewing and management facilities are not in themselves sufficient to ensure optimal project productivity. The number of tasks developers must manually perform with their tools, no matter how effective and efficient the tools are, continues to increase. The solution is provision of various forms of automation in the software tools developers use - the tools carry out perhaps a wide range of activities for the developer at appropriate times and inform the developer of results of actions in appropriate ways. We have focused in recent years on two areas of automation in software tools: (1) generating code from high-level software specifications; and (2) utilisation of high-level software information by agents to support collaborative work, change management and component testing. From our experiences developing a number of software tools using these automation approaches, we have learned a number of lessons for further research in these areas. These include: the need to support software tool meta-model extension; the need for on-the-fly enhancement of tool notations, event processing and code generation facilities; support for software artefact change propagation and annotation; the need to have reflective, high-level information to running software system components; and the continuing challenges of enhancing COTS tools with these kinds of automation facilities, including the need for sharable, extensible software information models for software tools and open tool infrastructure. We describe 3 software tools generating code from high-level descriptions (performance test-bed generator; data mapper; and adaptable user interface designer), and 3 tools providing event-driven agents (plug-in collaborative work components; requirements management tool; and deployed component testing agents).

Supporting Global Software Development with Event Notification Servers (C. De Souza, S. Basaveswara, D. Redmiles) Along wih the rapid globalization of companies, the globalization of software development has become a reality. Many software projects are now distributed in diverse sites accross the globe. The distance between these sites creates several problems that did not exist for previously co-located teams. Problems with the coordination of the activities, as well as with the communication between team members emerge. This paper describes how event notification servers are a useful technology for supporting global software development. They can facilitate the development of collaborative tools, which can be used to support distributed software development. In general, advantages of using event notification servers are described as well as specific problems that they help solve.

[1] http://www.isr.uci.edu/events/ASE-Workshop-2002/

Open Modeling in Multi-Stakeholder Distributed Systems: Model-based Requirements Engineering for the 21st Century (R. Hall) Multi-stakeholder distributed systems (MSDSs), wherein the constituent nodes are designed or operated by distinct stakeholders having limited knowledge and possibly conflicting goals, challenge our traditional conception of requirements engineering. MSDSs, such as the Internet email system, networks of web services, and the Internet as a whole, have globally inconsistent high-level requirements and, therefore, have behavior which is impossible to validate according to the usual meaning of the term. We can sidestep this issue by changing the problem from "does the system do the right thing" to "will the system do the right thing for me (now)?" But to solve that simpler problem, we need a way to predict behavior of the system on inputs of interest to us. OpenModel proposes to solve this by establishing open standards for behavioral modeling: each node will provide via http (or through a central registry) a behavioral model expressed in terms of shared domain-specific function/object theories. A tool will support validation by assembling these models and simulating, animating, or formally analyzing the assembled model, helping the user to detect unfavorable behaviors or feature interactions in advance. This paper presents the OpenModel proposal and discusses its potential advantages, challenges, and limitations.

Using te Semantic Web to Construct an Ontology Based Repository for Software Patterns (S. Henninger) Patterns, particularly design and usability patterns, have become a popular way to disseminate the current state of knowledge in certain software development issues. Many books have been written and people are using the pattern approach to encode knowledge ranging from management practices to risk assessment patterns. The continued explosion of patterns collections have caused a couple of clear problems. The first is the issue of quality and how one knows whether a pattern provides sound advice. The second is finding the right pattern for a particular problem. In this abstract, I propose the semantic web as a medium to start representing the relationships between patterns and track which are used most often or rated highly by peers. This approach not only supports the process of finding patterns, but also allows for the construction of agents that let developers know when a given pattern is applicable.

Managing Software Projects in Spatial Hypertext: Experiences in Dogfooding (F. Shipman) Managing long-term, research-oriented software projects requires more flexibility and open-endedness than most production-oriented software processes provide. We have been exploring the use of spatial hypertext to manage such projects. Spatial hypertext allows users to place information objects in visual spaces and use visual cues and spatial relations to represent inter-object relations. Over time, users develop a visual language to express characteristics of their task. The Visual Knowledge Builder (VKB), our particular spatial hypertext system, uses heuristics to recognize structure in user-generated layouts and includes navigable history for returning to earlier states of the spatial hypertext. This paper reflects on our experiences in dogfooding - using our own research prototype - for two projects for more than two and a half years and what these experiences might mean for using spatial hypertext in other software development contexts.

The TOBIAS Test Generator and its Adaptation to Some ASE Challenges (Y. Ledru) In the past decade, a scientific community has emerged around the notion of "Automated Software Engineering". This community has made several advances in two kinds of challenges: the complexity of processing software engineering information, and the difficulty to capture knowledge about software. This position paper first recalls these challenges. It then describes how these challenges influenced the design of the TOBIAS test generation tool.

Sleeping at Night: Perpetual Monitoring of Environmental Assumptions (S. Fickas, M. Skorodinsky, M. Feather) There is a paucity of methodologies for reasoning about an operational view of the environment of an embedded system. This is problematic given that critical failures of a system can often be traced to environmental behavior. We look at two aspects of the problem. First, we discuss a proto methodology for reasoning about "bad" environment behavior drawn from the security area. In particular, we suggest that work in analyzing crypto protocols might be a starting place to build environment modeling tools for the more general class of embedded systems. Second, we argue that one outcome of doing good analysis of the environment is the generation of environmental assumptions. That is, if a model discovers all of the ways that the environment of an embedded system can cause it to fail, it is unlikely that all such failures can be handled by the system. Using rational constraints on time and money, the system is designed and built with environmental assumptions: it is assumed that the environment will not show certain behavior during system operation. However, there is some good that can come from tracking assumptions made at analysis time down to runtime in the deployed system. We discuss means of doing this tracking and monitoring. We ground our work on a case study recently completed on a NASA software subsystem on a deep mission spacecraft.

Acknowledgments The workshop could not be organized without the contributions of various people from ISR's staff. In particular, we want to warmly thank Debra Brodbeck, Susan Knight, and Kiana Fallah, for putting together this "just-in-time" workshop.

References

[1] D. Redmiles. Proceedings of the 2002 Workshop on the State of the Art in Automated Software Engineering. Technical Report UCI-ICS-02-17, Department of Information and Computer Science, University of California, Irvine, june 2002.