January 2000

Software Engineering Tools

J. Gray
University of Wollongong

Software Engineering Tools

Abstract
Automated tools play an important role in the promotion and adoption of software engineering methods and processes. The development of these tools is itself a significant software engineering task, requiring a considerable investment of time and resources. There are a large number of different kinds of automated software engineering tools, variously known as CASE, CAME, IPSE, SEE, and metaCASE tools. Although these tools differ in the particular methods, activities, and phases of the software development cycle to which they are applied, constructors of these tools often face similar implementation issues. Decisions about host computing platform, implementation language, conformance with standards and reference models, choice of repository, integration and interoperability mechanisms, and user interface style have to be made. This mini-track is based around the experience reports of researchers and practitioners actively involved in software engineering tool development.

Disciplines
Physical Sciences and Mathematics

Publication Details
This paper originally appeared as: Gray, J, Software Engineering Tools, Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, 4-7 January 2000, 3300-3301. Copyright IEEE 2000.

Software Engineering Tools

Jonathan Gray
School of Information Technology and Computer Science
University of Wollongong, NSW 2522, AUSTRALIA
Tel +61 2 4221 3606, Fax +61 2 4221 4170
jpgray@computer.org

Abstract

Automated tools play an important role in the promotion and adoption of software engineering methods and processes. The development of these tools is itself a significant software engineering task, requiring a considerable investment of time and resources. There are a large number of different kinds of automated software engineering tools, variously known as CASE, CAME, IPSE, SEE, and metaCASE tools. Although these tools differ in the particular methods, activities, and phases of the software development cycle to which they are applied, constructors of these tools often face similar implementation issues. Decisions about host computing platform, implementation language, conformance with standards and reference models, choice of repository, integration and interoperability mechanisms, and user interface style have to be made. This mini-track is based around the experience reports of researchers and practitioners actively involved in software engineering tool development.

1. Background and motivation

The purpose of this mini-track is to bring together a community of software engineering practitioners and researchers who have an interest in developing software engineering tools. The mini-track should be of interest to anyone concerned with:

- tool construction technologies and techniques;
- development and application of new tools;
- evaluation of tools.

By *software engineering tool* we mean any software tool that provides some automated support for the software engineering process [1]. This is quite an encompassing definition that covers a number of levels of automated tool support, including:

- support for development activities, including specification, design, implementation, testing, and maintenance;
- support for process modeling and management;
- meta-tool technology, such as metaCASE products, used for the generation of custom tools to support particular activities or processes.

Within each level of support, we can find differing breadths of support [2]:

- individual tools that support one particular task;
- workbenches, or toolsets, that support a number of related tasks;
- environments that support the whole, or at least a large part, of the development process.

These definitions include many different kinds of software engineering tool variously known as CASE (Computer Aided Software Engineering), CAME (Computer Aided Method Engineering), IPSE (Integrated Project Support Environment), SEE (Software Engineering Environment), metaCASE, CSCW (Computer Supported Cooperative Work), and Workflow Management Systems.

The mini-track focuses on practical issues of the design, implementation, and operation of these tools, with the intention of sharing experiences and exchanging ideas so that our future tool development activities will be more productive and the tools more useful. The authors in this mini-track report on tool development covering a wide range of topics including metaCASE approaches, component based technologies, process modelling, repository organisation, distribution and configuration, data interchange, HCI/GUI, and cognitive and social aspects of tool development. Given this range of topics, it is hard to classify each paper into a single topic area. What follows below is a short overview of each paper and a brief description of the topics addressed.

2. Papers and topics

Understanding the cognitive processes involved in software development, and codifying knowledge about the software artifacts produced in this process, is an important and challenging undertaking. Encoding the experiences of software developers through the use of design patterns [3] is a topic explored in the paper by *Reiss*. The author presents a novel pattern language, and he describes the

PEKOE tool for assisting the identification, classification, creation, and maintenance of design patterns. The tool allows programmers to work with both design patterns and code simultaneously. Patterns can be saved in a library that accompanies the PEKOE system, and the patterns can be verified, maintained as the source evolves, and edited to modify the source.

Software engineering tools collect and store valuable amounts of information of various types including software designs, process management information, and meta-model data. To assist engineers in collaborative development work, these tools need to inter-operate and exchange information. Various classification schemes [4], reference models [5], and standards [6][7][8] have been proposed to tackle the problems of interoperability and data interchange. The paper by *St-Denis, Keller, and Schauer* examines the topic of data interchange in the context of a design recovery environment known as SPOOL. The authors describe the difficulties involved in model interchange and they evaluate a number of solutions to this problem. There is currently a lot of interest in this topic by standards organisations, and the new XMI format [9] looks like a very promising interchange format that may become widely adopted.

With the increasing popularity of distributed systems, there is demand for software engineering tools that support software engineering in a distributed manner, across a wide area, and possibly over heterogeneous networks [10]. *Lehto and Marttiin* examine the topic of collaborative working and the development of groupware tools to support this kind of activity. The authors describe theories of collaborative working, and they report their experiences with the with the Timbuktu system for supporting collaborative design.

The use of meta-tool technology is an important topic in software engineering tool development. The objective is to (re)build tools and tool components in a rapid manner and at the highest possible level of description. This topic is addressed in the paper by *Kahn et al*. The authors explore the generation of implementations of tool components, such as interchange formats, database schemas, and application program interfaces, from high level, implementation independent specifications. This work is focused on tools, based on the ISO STEP/EXPRESS standards [7] [8], for supporting major product manufacturing domains. The authors describe a transformation system, known as STEPWISE, for manipulating specifications written in EXPRESS, and they provide example transforms to illustrate this behaviour.

The manipulation of graphical representations of software artifacts is an important topic in software engineering tool development. The generation of new, customised, graphical modeling tools, tailored to domain-specific notational conventions, is the theme of the paper by *Sapia et al*. The authors describe their generic modeling tool, known as GraMMi, and they explain how it can be configured at run time to different notations by reading specifications of the desired graphical notation from a metadata repository. The incorporation of a four layer metadata framework, a layered system architecture, and a model-view-controler (MVC) user interface [11] are features of GraMMi that tool developers will find particularly relevant and interesting.

The generation of tools from high level specifications and the manipulation of visual representations of software are topics addressed in the paper by *Mernik et al*. The authors describe the LISA system, in which, formal language specifications [12] are used to generate language specific program development environments. This work addresses several important software engineering issues including: incremental development of new programming languages; software development using visual design languages; and the portability of the generation system and its tools across different computing platforms.

3. References

[1] Sommerville, I. *Software Engineering,* Addison-Wesley, (1995).

[2] Fuggetta, A. “A classification of CASE technology”, *IEEE Computer*, Vol 26, No 12, December (1993), 25-38.

[3] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. *Design Patterns,* Adison-Wesley (1995).

[4] Thomas,I. and Nejmah, B. "Definitions of tool integration for environments", *IEEE Software*, Vol 9 No 3, March (1992), 29-35.

[5] Wakeman, L. and Jowett, J. *PCTE: the standard for Open Repositories,* Prentice Hall, (1993).

[6] Electronic Industries Associates. "CDIF: CASE Data Interchange Format Technical Reports." CDIF Technical Committee, Electronic Industries Associates, Engineering Department, 2500 Wilson Blvd, Arlington, VA 22201, USA (1994).

[7] ISO 10303-11. Part 11: "EXPRESS Language Reference Manual", (1994).

[8] ISO 10303-21. Part 21: "Clear Text Encoding of the Exchange Structure", (1994).

[9] Object Management Group. "XML Metadata Interchange (XMI)", OMG Document ad/98-10-05, October (1998). Available from http://www.omg.org/docs/ad98-10-05.pdf.

[10] Agha, Gul A. "The Emerging Tapestry of Software Engineering", *IEEE Concurrency, Parallel, Distributed & Mobile Computing*, vol.5, no.3, July-Sept (1997), Special Issue on Better Tools for Software Engineering, pp.2-4.

[11] Lee, G. *Object-oriented GUI application development,* Prentice Hall, (1994).

[12] Wolper, P. "The meaning of "formal": from weak to strong formal methods", *International Journal on Software Tools for Technology Transfer*, Vol 1, No 1+2, (1997) 6-8.