Modeling Requirements for Future CASE: Modeling Issues and Architectural Consideration

PENTTI MARTTIIN
KALLE LYYTINEN
MATTI ROSSI
VELI-PEKKA TAHVANAINEN
KARI SMOLANDER
JUHA-PEKKA TOLVANEN

University of Jyväskylä

In this paper we discuss some requirements for future CASE (Computer Aided Software/Systems Engineering) environments. These requirements include increased modifiability and flexibility as well as support for process and agent models. We claim that they can only be addressed by developing more powerful representation and modeling techniques. As a possible basis for modeling various techniques, we outline a general information architecture for a future CASE environment. In addition, we propose primitive types for specifying techniques, the development process, and agent models, and use these types for modeling an example methodology and examine how the requirements are or can be supported in our example.

Interest in extending the functionality of Computer Aided Software Engineering (CASE) tools has been spurred up by the rapid advances in computing power, object-oriented database management techniques and graphical interfaces. As a result, numerous integrated tool and support environments have been launched in the recent years. In general, a CASE tool can be seen as a mechanism that empowers an information system (IS) developer through supporting a variety of development tasks by managing and manipulating different kinds of IS representations. As Forte and Norman (1992) point out, CASE tools have been successful in automating many routine software development tasks. The most common way to support development tasks using current CASE tools is to help in deriving graphical IS specifications and then transforming them into textual representations for correctness checking and reporting. Unfortunately, these tools fail to address several critical aspects of IS development (ISD) and have problems with their views of (meta) data, development process and group work support.

The surprisingly slow diffusion of CASE tools, especially front-end tools, and their low effect on productivity has caused confusion in tool developers and users. One reason for the low acceptance of CASE seems to be its weak support for the users’ native methods and methodologies. Because the selection of methodologies seems to follow some natural “evolution” and great differences exist between organizations, between system development projects, and even between individual developers with respect to dependency, scope and style of methods used, the methodology support in computer-aided environments should be flexible. Moreover, it should take into account various development contingencies, such as the nature of the system to be developed or the level of experience among the system developers. Such needs have been observed in empirical studies (Aaen et al., 1992; Orlikowski, 1988; Shapiro, 1991; Wijers and van Dort 1990).

Most current CASE tools lack customizing capabilities, support only a fixed set of methods, and thereby operate on a fixed metamodel. During the last few years, however, new CASE shells with changeable repository (meta data) schemas have appeared on the market. Thus, possibilities for building a customizable repository are already there. Examples of commercial CASE shells are VSF (Pocock, 1991) and Paradigm+. Other, research oriented CASE shells with similar features are RAMATIC (Bergsten et al., 1989), MetaEdit...
(Smolander et al., 1991), and MetaView (Sorenson, Tremblay and McAllister, 1988).

Recently, several attempts have been made to broaden the scope of CASE environments. Examples of these are the support of Computer-Supported Cooperative Work (CSCW) (Conklin and Begeman, 1988), business modeling and re-engineering (Spurr et al., 1993), strategy formulation and architecture specification (Chen, Nunamaker and Weber, 1989; Rose, Maltzahn and Jarke, 1992). Clearly, support for these tasks is needed if early phases of systems development will be covered by CASE. CASE researchers are also searching for solutions to specific technical design issues that are currently poorly supported in CASE, such as transformation support (Brinkkemper et al., 1989; Rossi et al., 1992) and version control (Katz, 1990; Hahn, Jarke and Rose, 1991). Object-orientation (Booch, 1991; Rumbaugh et al., 1991) and supplementing CASE environments with hypertext features (Garg and Scacchi, 1990; Cybulski and Reed, 1992; Kerola and Oinas-Kukkonen, 1992; Oinas-Kukkonen, 1993) have been proposed as solutions to some of these problems.

In general, CASE environments address the obvious need to produce, analyze and manage descriptions of various kinds during interactive system development (ISD). This need can be viewed as a metamodeling problem: what are the requirements for languages to describe and model the data stored, represented and manipulated in the IS repository, and what is the process to specify the content and functionality of such a repository? We define metamodeling to be the task that produces a metamodel for a specific ISD situation taking the contingencies into consideration. Accordingly, a metamodel embraces a methodology specification tailored to a specific project. In general, this issue has been largely ignored until recent years, and commercial CASE tools suggest only ad hoc solutions to this problem. The goal of this paper is to specify the requirements for the information architecture of future CASE (shell) environments to help address these problems in a more systematic manner.

The paper is organized as follows. First, we take a look at the future and try to characterize an ideal CASE environment at the turn of the millennium. Also, we specify the elementary requirements for representing methodology data in such an environment. After that, we present a general information architecture, show an example of the architecture in use and discuss how to meet the requirements within an integrated environment. Finally, we draw some conclusions about the desired level of methodology support in future CASE environments.

“2001: A CASE odyssey”

To give an idea of a future CASE environment we outline here a scenario of a session using such an environment:

Think of a usual morning in the software development center of NanoSoft Corp. A software engineer and project manager S.E. logs into her Beta+ workstation and picks up a “CASE Factory” icon in the “WorkSpace manager”. The “CASE Factory” icon starts the CASE environment’s desktop window. It shows as icons her current projects and the tools she has picked for use. She then sets up the environment for a certain project, in which a graphical “view manager” provides tools to modify or look at the different parts of the IS specification. She begins to work by modifying the contents of a project’s schedule. The schedule comes up as an activity net in a graphic window. Being more comfortable with textual definition of tasks, she changes the type of the window from graphic to text and continues her work with a task list. After having modified the project she looks at the electronic white board where one of her project members notifies her about a problematic OER-design. She closes the schedule window and clicks the ““OER-diagram”” icon, causing the OER-diagram to appear, and locates the problematic part by searching for the note assigned to it. She sketches a solution for the problem by backtracking the design into an earlier version and then starts a negotiation session to converse about the solution with other project members. An automatic repository agent also takes part in the discussion by showing those parts of the project that are affected by the design change. When the others have accepted the solution, she stores the new design into the repository and continues with her other duties. Meanwhile, the repository agent will check and annotate all affected documents and possibly update some transformations and documents automatically.

As a project manager, S.E. uses the same shell with special reporting tools to monitor the state of the project and to estimate the cost and risks of the project. When e.g. she needs the FPA++ cost analyzing software she can initiate it from within the environment, because the FPA++ tool conforms with the “common software platform” standard, and the local CASE environment manager has attached it to the CASE repository allowing FPA++ to use the repository services. As S.E. performs these monitoring operations repetitively, she has specified semiautomatic and automatic agents that perform routine tasks (e.g. collect managerial data, check the consistency of IS specifications, do library management and notify of slippings from the schedule) for her, in a manner similar to the assistants in the Programmers’ Apprentice (Rich and Waters, 1990).

When the users of the CASE environment find a modeling technique inappropriate, the CASE environment manager can use the CASE administrator’s toolset (Martin, 1988a) to modify the repository in order to overcome the problems. The toolset gives the CASE environment manager the means to meet the tool users’ different needs such as organizational standards and the users’ modeling experience. It gives her the opportunity to reconfigure tools and the repository to capture
Related Content

Embracing Change: How South Ayrshire Council Library Service Became a World Leader in Electronic Resources Management
www.igi-global.com/chapter/embracing-change/90183?camid=4v1a

Information Technology/Systems Offshore Outsourcing: Key Risks and Success Factors
*Journal of Information Technology Research* (pp. 72-92).
www.igi-global.com/article/information-technology-systems-offshore-outsourcing/3693?camid=4v1a

ICT in Medical Education in Trinidad and Tobago
www.igi-global.com/chapter/ict-medical-education-trinidad-tobago/22795?camid=4v1a

Direct-to-Consumer Genetic Testing: Interdisciplinary Crossroads
www.igi-global.com/article/direct-consumer-genetic-testing/69508?camid=4v1a