A Viewpoint-based Framework for Software Development Environments[1]

Anthony Finkelstein, Jeff Kramer & Bashar Nuseibeh[2]

Abstract

This paper outlines a framework for "CASE tool" development and integration which supports distribution.

Motivation

The development of most large and complex systems necessarily involves many people - each with their own perspective on the system defined by their skills, responsibilities, knowledge and expertise. This is particularly true where the system is a composite system, that is one which deploys a variety of different technologies (software, hardware, mechanical and so on). Inevitably, the different perspectives of those involved in the process intersect and overlap, giving rise to a requirement for coordination. The intersections are, however, far from obvious because the knowledge within each perspective is represented in different ways. Further because development may be carried out concurrently by those involved, different perspectives may be at different stages of elaboration and may each be subject to different development strategies.

The problem of how to guide and organise development in this setting - many actors, sundry representation schemes, diverse domain knowledge, differing development strategies - we term "the multiple perspectives problem".

Tool integration, the central problem in software development environments, can be viewed as a special case of the general multiple perspectives problem. Support for managing, ensuring consistency and transferring information between many tools with different roles, maintaining different models and which are used to support concurrent and distributed work depends on support for structuring and organising multiple perspectives.

Viewpoint Framework

We have developed a framework which supports the use of multiple perspectives in software development. The primary building blocks of this framework are "viewpoints".

A viewpoint can be thought of as a combination of the idea of a "actor", "knowledge source", "role" or "agent" in the development process and the idea of a "view" or "perspective" which an actor maintains. In software terms it is a loosely coupled, locally managed object which encapsulates partial knowledge about the system and domain, specified in a particular, suitable representation scheme, and partial knowledge of the process of development.

Each viewpoint is composed of the following components, which we call slots:

a representation style, the scheme and notation by which the viewpoint expresses what it can see;

a domain, which defines that part of the "world" delineated in the style;

a specification, the statements expressed in the viewpoint's style describing particular domains;

a work plan, describing the process by which the specification can be built;

a work record, an account of the history and current state of the development.

[1] Presented at IEE Colloquium on "Architectures for Distributed Development Support Environments", 4th November 1991, Savoy Place, London, Digest Number: 1991/162.

[2] Department of Computing, Imperial College, 180 Queen's Gate, London, SW7 2BZ, Email: {acwf, jk, ban}@doc.ic.ac.uk.

A number of viewpoints may employ the same style and the same work plan to produce different specifications for different domains. We therefore define a reusable viewpoint template in which only the style and work plan slots are elaborated. A single viewpoint template may be instantiated to yield several different viewpoints, and by extension several specifications. A method in this context is a configuration of viewpoint templates.

Integration is achieved by checks maintained locally within each viewpoint and enforced, where it is required, by the work plan. These checks define partial consistency relations between the different representation styles. Consistency is checked incrementally between viewpoints at particular stages rather than being enforced as a matter of course. The checks may be used to determine whether viewpoints are consistent with each other and as transformations to move information between viewpoints. A full account of the framework is given in Finkelstein, Kramer & Goedicke 1990.

Implications

Unlike existing frameworks for tool integration based on repositories, in which all tools access a common object base, and selective broadcast mechanisms, in which updates are selectively broadcast to tools registered with the message server, the viewpoint framework is loosely coupled and fully distributable. Experience suggests that centralised environment architectures based on existing frameworks are difficult to change and are not readily scaleable. Further, such centralised architectures suffer from inherent performance bottlenecks.

Conventionally representation schemes, specifications and process models are artificially separated. The viewpoint framework packages them together in a way which more accurately reflects the organisation of software development.

Status

The framework offers two areas for automated support. On one hand, tools are required to facilitate method development, description and integration. On the other hand, tools are also required to support method use within the viewpoint framework. This includes tool support for individual viewpoint development and the construction of specifications, obtaining method guidance and dynamically creating viewpoints as development proceeds. Further support is required for management and navigation across viewpoints.

The √iewer is a prototype environment supporting the framework. Implemented in Objectworks/Smalltalk V4.0, it supports method development and integration, facilitates the construction of individual viewpoint tool support, and allows the smooth transition to, and manipulation and management of, viewpoints during method use.

The √iewer has been used to provide support for substantial fragments of some existing software development methods. These include CORE (Mullery 1985) and the Constructive Design Approach (Kramer, Magee & Finkelstein 1990). Further work on methods based on complex formal representation schemes is continuing. We are currently investigating notations which can be used to define viewpoint templates and are investigating distribution of the √iewer within the REX environment (Kramer 1990).

References

Finkelstein A. Kramer J. & Goedicke M. (1990); “ViewPoint Oriented Software Development”; Proc. of 3rd International Workshop Software Engineering & its Applications; Cigref EC2 V1, pp337-351.

Kramer, J. (1990); “Configuration Programming - A Framework for the Development of Distributable Systems”; Proc. of IEEE Int. Conf. on Computer Systems and Software Engineering (CompEuro 90), Tel-Aviv, Israel, May 1990, pp374-384.

Kramer, J. Magee, J. & Finkelstein, A. (1990); “A Constructive Approach to the Design of Distributed Systems”, Proc. 10th IEEE Int. Conf on Distributed Computing Systems, Paris, June 1990.

Mullery, G. (1985); “Acquisition - Environment”; (In) Paul, M. & Siegert, H. “Distributed Systems: Methods and Tools for Specification”; Springer Verlag LNCS 190.