Exploiting MOOs to Provide Multiple Views for Software Process Support

Fabrizio Becattini Elisabetta Di Nitto Alfonso Fuggetta
Giuseppe Valetto

CEFRIEL – Politecnico di Milano

Via Fucini, 2 I-20133, Milano

Tel: +39-2-239541 Fax: +39-2-23954254

e-mail: {becattin, valetto}@cefriel.it, {dinitto, fuggetta}@elet.polimi.it

July 26, 1999

1 Introduction

Process-centered Software Engineering Environments (PSEEs) guide human beings in the execution of the structured, cooperative activities pertaining to the complex domain of software development. PSEEs exploit some form of *process model*. A process model is a formal description of the various steps that must be carried out by *process agents*, i.e. members of the software development organization in order to pursue its objectives.

A critical problem for PSEEs is the definition of a proper interaction metaphor between the system and the users. In some cases, PSEEs are not fully accepted in industrial contexts because they are perceived to be too prescriptive and too limiting of the freedom of human actors. The user interaction metaphor plays an important role in the establishment of this perception. To make a system successful, this metaphor has to be intuitive and easy to grasp. Moreover, it must provide human actors with an effective visualization of the state of the software process. In other words, it has to let human actors fill *process aware*. Indeed, the user interaction environment should be flexible enought to support all the viewpoints different users have on the process [1, 2].

As discussed in [6], MOOs [3, 8] can provide interesting mechanisms to define user interaction metaphors for PSEEs. MOOs are Object-Oriented programming and execution environments that support the creation of *virtual environments* where multiple users can interact. MOOs have been designed for entertainment and socialization within connected and distributed communities of heterogeneous users. Indeed, they enable the definition of a very intuitive metaphor also for other kinds of cooperative applications [7, 5]. Within a virtual environment, users can move in a set of virtual rooms interconnected through doors and containing objects. Users are allowed to explore the contents of rooms, create and manipulate objects, contact other users who are visiting the same room, and – under certain conditions – extend and enrich the virtual environment as a whole.

By mapping the important concepts of processes (tasks, roles, artifacts, relationship among tasks) into rooms, objects, and the other elements typical of MOOs, it is possible to create a virtual environment in which process agents can accomplish tasks – either autonomously or cooperating with other agents – in an intuitive way.

In this paper we present our experience in using MOOs to implement a prototype of a PSEE that offers multiple views over the enacted software process; since all the views are constructed as portions of a virtual environment, they present particularly intuitive interaction metaphors to the users of the PSEE.

2 Developing a PSEE by Exploiting MOOs

2.1 Mapping software process concepts onto MOOs

The most fundamental issue in developing a user interaction metaphor based on MOOs for a PSEE is certainly the definition of a proper mapping of the basic concepts found in processes (artifacts, tasks, roles, etc.) into the main elements of MOOs (rooms, objects, characters, etc.).

A first possible mapping is the one proposed in PROMO [6], in which rooms in the MOO represent tasks in the process, the connections among rooms represent the precedence relationships that hold between tasks, and the objects that can be located in a room represent either the artifacts that are being used or created, or the tools used to operate on artifacts. The user can navigate through the resulting virtual environment, can create and modify process artifacts by exploiting the tools available in each rooms, and can move process artifacts through the rooms. All movements are regulated by a number of constraints that are checked each time an item (either users or artifacts) passes through a door. The constraints implement the data and control flow rules of the given process. We call this way of representing the process *task–centered*. We argue that a task–centered representation, while on the one hand clearly shows to process agents the logical structure of the process in terms of its tasks, on the other hand is not sufficient to provide a high level of process awareness, since it fails to cover other important viewpoints over the process. For instance, it does not convey information about the assignments of process agents to the various tasks and artifacts. Moreover, it does not show the structure and the state of the product being developed, with its various components.

We envisage that to augment process awareness two other mappings can be added to a task–centered view over the process, resulting in views that focus on those other important aspects: the *workspace–centered view* and the *artifact–centered view*. These mappings have been also discussed in [6] in a preliminary way.

In the workspace–centered view, the rooms in the virtual environment represent the workspaces where process agents work. A workspace contains all the artifacts the corresponding human agent (or a collaborating group of agents) is working on, together with the tools employed. Artifacts and tools are collected on top of desktops within the workspace, and each desktop is associated to a task to be executed. The connections among rooms represent the organizational structure of the project team. The constraints attached to the entrances of workspaces can be used to enforce specific privileges and permissions, according to the *user roles* defined in the process.

In the artifact–centered view, rooms and connections among rooms represent the breakdown structure of the product being developed, thus emphasizing the composition relationships between the various product parts. In particular, MOO rooms represent nodes of the product breakdown structure (i.e. a part of the overall product) and exits leading from room to room correspond to the connections among nodes (i.e. the composition relationships between product parts). The rooms can also be used to store the various versions of the same software artifact. For space reasons we do not further discuss about this view in the rest of the paper.

2.2 Building the Process and its Views

Our prototype exploits a MOO system to construct a virtual environment where a software process is both modeled and enacted. To specify a process model the *process engineer* enters in a special room called Process Repository and exploits the tools associated with that room to define tasks, artifacts, precedence relationships among tasks, roles covered by the human agents participating in the process, tools and resources used in the process. The operations performed by the process engineer result in the population of the Process Repository with process data and in the generation and furnishing of new sets of rooms that define the three views outlined above.

Figure 1 shows a snapshot of the virtual environment created for an interactive code inspection process (see [9]). All the elements appearing in each virtual environment fragment corresponding to a view are MOO objects that constitute the *images* of some process description objects kept in the Process

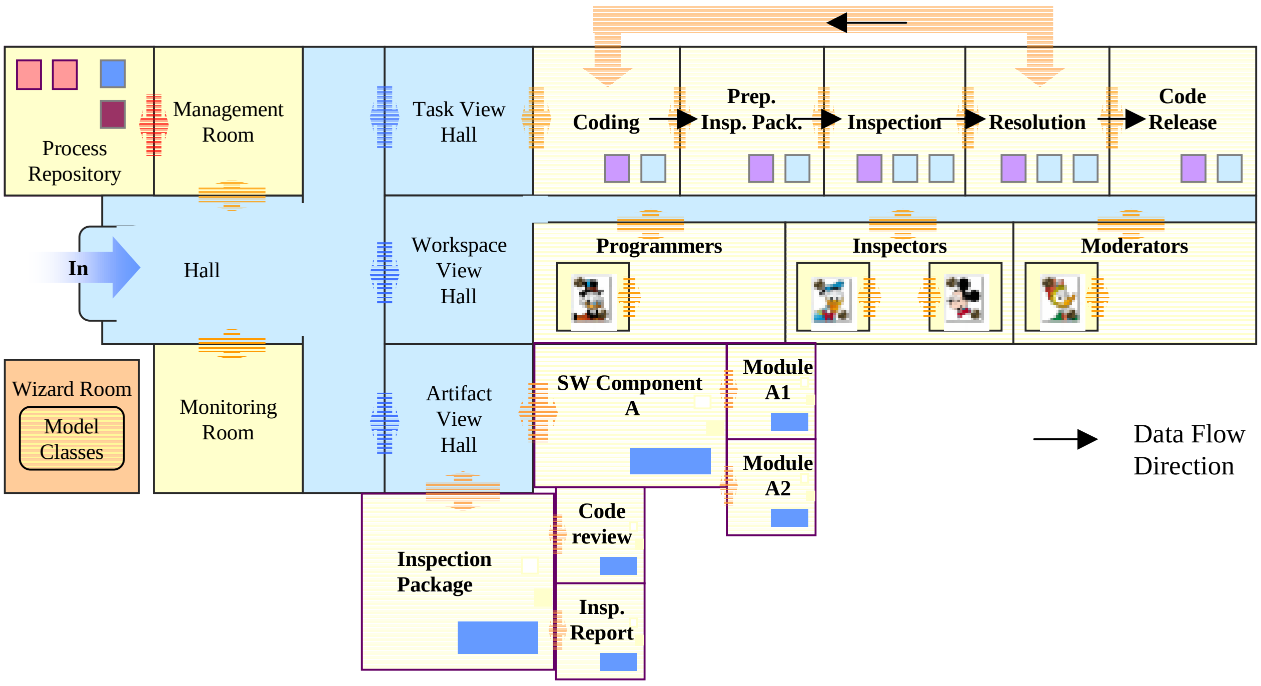

Figure 1: *A snapshot of the virtual environment for the code inspection process.*

Repository. The image objects within each view provide to human agents information about the process and its state in accord with the interaction metaphor associated with the view.

2.3 Executing the Process

In any PSEE, when the process is enacted, new task instances are created and executed, roles are assigned to the project team members, product artifacts are created and manipulated, etc. In our system, this is reified as a modification of the virtual environment. In some cases this modification affects even the topology of the virtual environment. For instance, a new room is created in the task-centered view when a new task is instantiated, or in the workspace-view when a team member is assigned to a role. This evolution of the virtual environment topology is necessary to keep track of the dynamicity of the process. However, it is also necessary to keep distinct the process model defined at design–time from the process enacted at run–time (i.e. to capture both the *type* and *instance* levels of the process description).

We have resolved this problem by introducing the concept of *inner* room. In particular, our prototype creates new rooms representing instances of tasks inside the room representing the corresponding task or or workspaces for new users in the room for the corresponding role. This guarantees that changes in the topology of the virtual environment are kept local to each outer (type) room and are not confusing to human agents whenever they want to focus just on the static structure of the process model. Figure 2 shows an example in which rooms `Inspection` and `Resolution` contain two and one inner rooms, respectively. Those represent different instantiations of the `Inspection` and `Resolution` tasks that operate on different artifacts. The connection between `Inspection` and `Resolution`, which represents a precedence relationship between the corresponding tasks, is also reified at the instance level through a connection between `I1` (that represents the inspection of module `moduleA.c`) and `R1` (that represents the resolution meeting corresponding to the same module).

Another important aspect related with the enactment of the process is that in a multi–view environment like ours all data and images must always be kept mutually consistent. During process enactment,

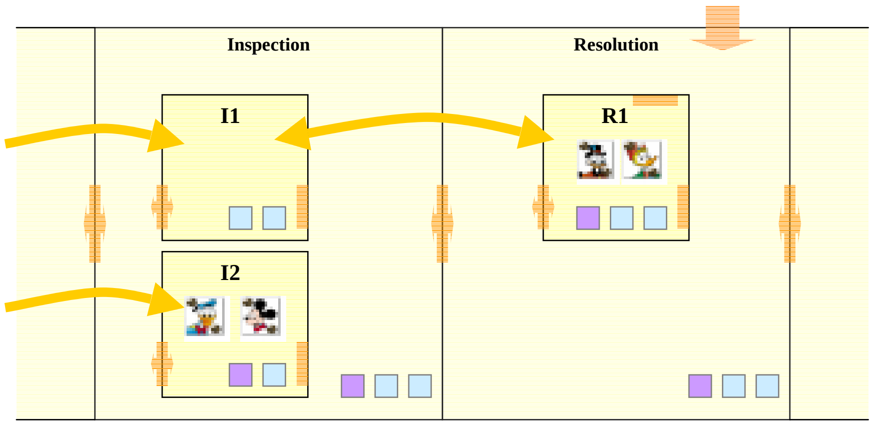

Figure 2: *Details of the Inspection and Resolution tasks.*

whenever a human agent performs an operation upon an image object from within a view, this operation is delegated to the corresponding object in the Process Repository, thus ensuring that any variation in the appearance, location or behavior of an image is properly reflected in the state of the Process Repository data and propagated to the corresponding images in the other views. For instance, if a user walks into an inspection room and starts operating on the artifacts contained in that room, not only the state of all concerned objects in the Process Repository are updated; also the workspace–centered view is automatically changed, since the system places a new desktop in the office of that user and puts on that desktop all the artifacts he/she is working on as well as necessary tools and resources to carry out that work.

A longer discussion about the interaction metaphors of the views and inter–view consistency aspects is presented in [4] and will be the subject of a further publication.

3 Discussion

Our research has produced a prototype that exploits MOOs for the description and support of process-centered environments according to multiple viewpoints. The immersive nature of a MOO–based PSEE, coupled with the ability to offer a number of different viewpoints over the process are the most important beneficial aspects we notice.

MOO systems, however, present a number of intrinsic hindering limitations. First of all, existing MOOs do not provide mechanisms for managing multiple views. A more sophisticated implementation of the MOO database, supporting natively standard viewing mechanisms (as those offered by other OODBMS like O2 [10]) would be clearly beneficial and would avoid the need to implement views on purpose – like we had to do – by replicating objects and developing ad hoc consistency mechanisms.

MOOs are – by their very nature – eminently self–contained systems. All the tools employed by users to carry out some work are usually defined, implemented and operated within the boundaries of the MOO programming and execution environment. In order to support the work of the users in a natural and effective way, an appropriate and exhaustive integration framework for legacy tools must be conceived. For example, PROMO suggests to exploit the MOO client–server architecture and to build special MOO clients that work as bridges or *proxies* between the MOO core and external applications. This represents a low–level control integration mechanism. Also, it might be useful to provide MOOs with some facilities to access data produced by other applications (data integration). Compliance of MOO systems to middleware standards – which is nowadays a common interoperability solution – should also be carefully analyzed and evaluated.

Working within a MOO is also often not as easy and natural as one would expect, because of the lack of adequate user interfaces. Originally, virtual environments offered simple textual user interfaces.

Then, as an effect of their growing popularity, numerous user–friendlier interface paradigms have been conceived and explored, leading to numerous graphic, hypermedia, multimedia, Virtual Reality (VR) and WWW–based systems. However, even with these advancements, the Human–Computer Interaction mechanisms of MOOs are still somewhat disappointing: for example, VR interfaces tend still to be too demanding with respect to client resources. Hypermedia interfaces, instead, are eminently static, i.e. do not support automatic notification to the client whenever a change occurs in the environment.

Our current research work is focused on reengineering our proof–of–concepts prototype to overcome the limitations of the MOO platform upon which we based our experimentation. In this reegineering activity we are also evaluating the possibility of superimposing the generic virtual environment interaction metaphor as well as the mappings we have devised for our process views upon existing commercial process support systems.

References

[1] A. Fuggetta L. Jaccheri. Dynamic partitioning of complex process models. Technical report, NTNU, Trondheim (Norway), 1998.

[2] D. Avrilionis P.Y. Cunin C. Fernstrom. Using views to support evolution and resue of software processes. In *The 18th International Conference on Software Engineering*, Berlin, Germany, 1996. IEEE Computer Society.

[3] R. Bartle. Interactive multi-user computer games. Technical report, 1992. ftp://ftp.lambda.moo.mud.org/pub/MOO/papers/mudreport.ps.

[4] F. Becattini. Ambienti virtuali come rappresentazione e supporto di processi collaborativi. Master's thesis, Università degli Studi di Pisa, 1998. In Italian.

[5] P. Curtis and D. Nichols. MUDs grow up: social virtual reality in the real world. In *The Third International Conference on Cyberspace*, XEROX Palo Alto Research Center, May 1993.

[6] J.C. Doppke, D. Heimbigner, and A.L. Wolf. Software process modeling and execution within virtual environments. *ACM Transactions on Software Engineering Methodology*, January 1998.

[7] S.M. Kaplan, G. Fitzpatrick, T. Mansfield, and W.J. Tolone. MUDdling through. In *the 20th Annual Hawaii International Conference on System Sciences*, Washington D.C., 1997. IEEE Computer Society.

[8] M. Keegan. A classification of MUDs. *Journal of MUD Research*, 2(2), 1997. http://journal.tinymush.org/v2n2/.

[9] J.M. Perpich, D.E. Perry, A.A. Porter, L.G. Votta, and M.W Wade. Anywhere, Anytime Code Inspections: Using the Web to Remove Inspection Bottlenecks in Large Scale Software Development. In *the 19th International Conference on Software Engineering - ICSE.* ACM, 1997.

[10] O_2 Technology. *The O_2C Reference Manual.* O_2 Technology, 1996.