The Need for a Cooperative Model: The Adele/Tempo Experience

Noureddine Belkhatir
LGI
BP 53
38041 Grenoble France
e-mail: belkatir@imag.fr

Walcélio L. Melo
University of Maryland
UMIACS,
College Park, MD, 20742 USA
e-mail: melo@umiacs.umd.edu

Position

Our position is that the software process is fundamentally cooperative. We have been developping concepts and mechanism to tackle with cooperation in the Adele/Tempo project.

Firstly, we integrated into the Tempo software process formalism a new concept: cooperation classes. In addition, we provided temporal, event-condition-action (TECA) rules in order to monitor and control cooperative activities. Based on TECA rules and cooperation classes, we can describe exchange information policies between software developers as well as rules about the organization, evolution and consistency of software projects.

Secondly, we have been developing a transaction management system which is able to support the Tempo cooperative model. This system has been built up to deal with un-predicted events provoked by activities carried out in cooperative work environments.

Overview of Adele/Tempo concepts

In our previous papers, we presented the process modeling strategy Tempo [2], which uses the Adele database as a resource manager. Tempo is an approach where activities and resources are represented as active objects. Each software process step is modeled as an active object which encapsulates operations, a collection of tailored resources and recursively, other process steps. Activities executed by humans are modeled as methods associated with processes and resources. Rules are used in order to control, order and monitor activity execution.

Software processes are performed by humans in Working Environments (WE) according to policies defined in a software process type. In fact, a work environment is the "implementation" of a software process type defined in a software process schema. As discussed in [1], software engineers are allowed to work in parallel.

Using the role concept [2], each WE can customize the characteristics of manipulated resources (attributes) and the activities on such resources (methods and rules), thus providing a kind of support for dynamic points of view [3]. In this way, the behavior of a resource object depends on the role it is playing in a WE. Fundamentally, a software process model is treated as a set of asynchronous and parallel WE's. This approach nicely combines the procedural [6] and declarative paradigms [5] where objects and human interactions are described structurally, by means of attributes and process types, and functionally, by means of methods and rules controlling the execution of such methods.

Rules are not simply used to control method execution. In fact, rules explicitly describe how software activities can or cannot be enacted (prescriptive and proscriptive knowledge) by humans and automated tools.

Discussion

We have splitted Tempo in two layers in order improve the way Tempo supports cooperation in a multi-user software development environment (see figure1).

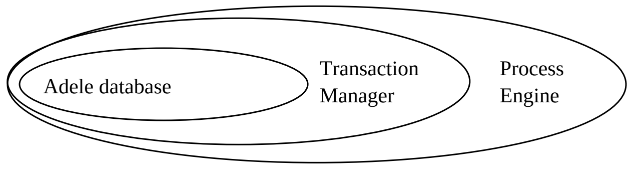

Figure 1: Conceptual architecture of Tempo.

1. Tempo process engine layer. Tempo process engine is responsible for monitoring the software process enacting in each single WE. It is also responsible for dealing with pre-defined cooperation policies. That is, cooperation scenarios can be defined and supported by the Tempo process engine using the concept of sensitive-connections, process type and object with roles. However, when an un-predicted cooperation situation (cooperation break-down) occurs, for example two work environments try to update a shared object, the transaction management layer is called to monitor the negotiation between conflicting WE's.

2. Tempo transaction manager layer. This layer is responsible for preventing/monitoring/resolving

conflicts when a shared object is accessed by cooperative WE's.

To clarify these two layers. Consider a simple scenario of cooperation in a software process engineering environment composed of only two WE's. A user is developing the interface of a module whereas another one is developing the body. Each user works in a different WE.

- At the process layer, we define for each process what activities should be performed and under what constraints and how these activities can normally cooperate using Tempo process programming language. For instance how to compile, test the module.
- The transaction manager is in charge to control situations where the development of the body need to be synchronized with the development of the interface to prevent the development of an old version of a body while a new version of interface has been developed.

Object-Oriented technology and Tempo

Object-oriented (O.O.) technology is suitable for our needs because it nicely integrates three important dimensions:

- What an object has (its composition).
- What an object is (inheritance).
- And how an object behaves.

The O.O. approach facilitates hierarchical decomposition of activities and the modeling of relations between data. Each activity in our model is a collection of conceptually related entities. Each WE has an owner. The WE determines:

- The representation of structural aspects of related entities by means of objects extended by roles.
- The scope of rules controlling the process enactment (WE creation and resource allocation to it) and performance in a WE.

To integrate cooperation explicitly, we extend the Tempo process programming language with:

1. A complex object describing the structural and functional aspects of the communication between WE's – the connection concept. A connection is an active object linking two WE's. A connection type describes the template for communication between WE's, i.e., (situations such as communication establishment and break-down, circumstances for collaboration and collateral effects of collaboration). These details are described by a special kind of rule (plug-on, plug-off and active-rules) and are supported by a trigger mechanism integrated into a broadcast management system.
2. Rules for controlling the communication and coordination between processes. Human interactions are driven and controlled by event-condition-action rules. Such a rule includes a pre-condition for interactions, the effects caused by interactions (event) and the interaction itself (action).

Temporal rules and Tempo

Considering the change evolution scenario presented in [4], we identify time as a primary component in supporting user interaction and cooperation. Thus, our research is concerned with the conceptual description of time. We incorporate the time dimension by extending the event-based rules. This results in a temporal, event-based rule processing approach. At the level of the database system, major attention is dedicated to the problem of managing temporal information by storing past history states.

ON *logical expression combining present and/or past events*
DO *Action*

Using temporal rules:

- The control of user interactions is facilitated because we can distinguish between completed and future interactions.
- User actions is better monitored because we can determine which activities have been carried out in a WE and the results produced by those activities.
- The management of user activities can be improved because the audit process is facilitated.

Conclusion

The main result of our approach is the view of the software process as a cooperative system where cooperation needs to be considered as a separate conceptual unit.

The necessity to model cooperation as a separate conceptual unit in our approach has been highlighted in the development of the Tempo system. The explicit definition of communication is a major support of cooperation. We can summarize the main results of this work as the following:

- The adaptation of O.O. techniques to capture human interactions;
- The adaptation of event-based rule processing to control the enactment and performance of software processes by humans and automated tools;
- The necessity of a conceptual framework in which the time dimension can be captured. This fact has led us to propose temporal-event-based rule processing for the management of temporal information in the database system;
- The necessity of treating cooperation as a top level object on the conceptual level.

Referring to our experience in developing tools and environments for programming-in-the-large, we have argued that the software processes are mainly cooperative. So far, most of the work of the process community has focused on architectural aspects, (i.e. cooperation is considered as a way to decompose and schedule activities.)

This position paper argued for the necessity to introduce cooperative concepts at a formal level. We have proposed:

- The concept of programmable and active connection where cooperative activities can be described.
- Temporal-rules which make it possible to control user activities and interactions.

Our long term objective is to build a framework that identifies relevant parameters for evaluating and measuring the performance and cooperation in a software process multi-agents environment. We are only at the beginning of this challenging task.

Acknowledgements

We would like to express our thanks to Gianluigi Caldiera, Carolyn Seaman and Roseanne Tesoriero for suggesting substantial and helpful revisions to the original text.

References

[1] N. Belkhatir and W. L. Melo. Supporting software maintenace processes in Tempo. In *Proc. of the Conf. on Software Maintenance*, pages 21–30, Montreal, Canada, September 1993. IEEE CS Press.

[2] N. Belkhatir and W. L. Melo. Evolving software processes by tailoring the behavior of software objects. In *Proc. of the Conf. on Software Maintenance*, Victoria, Canada, September 19934. IEEE CS Press.

[3] A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedicke. Viewpoints: a framework for integrating multiple perspectives in system development. *Int'l Journal of Software Engineering and Knowledge Engineering*, 2(1):31–57, 1992.

[4] M. I. Kellner, P. H. Feiler, A. Finkelstein, T. Katayama, L. J. Osterweil, M. H. Penedo, and H. D. Rombach. ISPW-6 software process example. In M. Dowson, editor, *Proc. of the First Int'l Conf. on the Software Process*, pages 176–186, Redondo Beach, CA, October 21–22 1991. IEEE CS Press.

[5] D. E. Perry. Policy-directed coordination and cooperation. In I. Thomas, editor, *Proc. of the 7th Int'l Software Process Workshop*, San Francisco, CA, October 16–18 1991. IEEE CS Press.

[6] M. Suzuki, A. Iwai, and T. Katayama. A formal model for re-execution in software process. In L. Osterweil, editor, *Proc. of the 2nd Int'l Conf. on the Software Process*, pages 84–99, Berlin, Germany, February 1993. IEEE CS Press.