

Requirements And Early Experiences In The Implementation Of The SPADE Repository

Sergio Bandinelli * Luciano Baresi Alfonso Fuggetta Luigi Lavazza
CEFRIEL and Politecnico di Milano[†]
Via Emanuelli 15, 20126 Milano (Italy)

1 Introduction

SPADE is a research project carried out at CEFRIEL and at Politecnico di Milano with the goal of developing an environment for Software Process Analysis, Design and Enactment. SPADE is centered on a process modeling language called SLANG (SPADE LANGuage), an extension of a high-level Petri net notation. In SLANG tokens represent objects, places are object containers and transitions are associated with a guard and an action that describe token transformations [1, 2].

The main features of SLANG are the following:

- SLANG is formally defined and thus can be automatically interpreted in order to support both the analysis and the enactment of a process model.
- A SLANG specification is composed of different model fragments, called activities. Activities are concurrently executed by multiple *process engines* and may communicate by means of shared data structures (namely, shared places).
- All the information describing a SLANG specification and its state is stored in a centralized repository. In particular, the repository is built on the top of an Object Oriented Data Base system.
- It is possible to change parts of a SLANG specification while the process is being enacted, in order to support process evolution.

In this paper we identify some requirements that a database system should offer to support these issues, and discuss our experience with a commercial OODB system.

2 Requirements for the SPADE repository

Many authors have already addressed the problem of identifying the requirements for a database supporting a software engineering environment. The early work by Bernstein [3] raises many general issues such as dimensions of objects, multiple data representations, flexible type system, versioning, long-lived transactions, etc. More recently, other papers have addressed the issue of the suitability of OODB systems as a vehicle to implement software engineering repositories (see for example [5]). In particular a first valuable experiment in using object-oriented databases to support the software development can be found in [6]. Most works emphasize the problem of object granularity and its implications in version management, long transactions, etc. It is our opinion that other quite strong requirements to database systems come out when we consider the problem of process evolution.

The SLANG interpreter uses an OODB to store both the process data manipulated by the process model (such as documents, code, tests data, etc.) and the description of the process model itself. Thus, process enactment involves also process evolution, since not only process data, but also the process model itself, may be modified during enactment.

We may see the SPADE repository as organized in two levels, the schema level and the instance level:

- The **schema level** contains the (meta) information about type¹ definitions. A part of these types is fixed, since they may not be modified by process enactment. This part corresponds to the definition of process language constructs, which are common to all process models. The other part

*Sergio Bandinelli is partly supported by DEC
[†]Contact information: Tel: +39-2-66100083,
Fax: +39-2-66100448, E-Mail: alfonso@mailier.cefriel.it.

¹With the word type, here we make reference to ADT definitions, providing the type structure definition and the corresponding operations to access this structure. In some OODB systems this concept may correspond to the keyword class.

describes the templates of the objects, manipulated in a specific software process model. This second group may be modified as a consequence of changes to the process model.

- At the **instance level**, we have two different sets of objects as well. The instances of the types in the fixed part of the OODB schema correspond to a specific process model definition (i.e., a collection of arcs, transitions, and places constituting a SLANG specification). The instances of the modifiable part of the schema correspond to process data (e.g., documents, source code modules, specifications, etc.) produced or modified during process enactment.

Summing up, we have the following scenario as described in Figure 1. It is not possible to change the definition of the SLANG language (fixed part of the schema). Changes to the variable part of the schema and to the instances of the fixed part correspond to changes in the process model definition. Changes to the instances of the modifiable part correspond to changes in the state of the enacted process model.

From the above discussion we can derive a first set of general requirements:

1. The process engine should be able to interpret any process model and thus can not depend on the types of the modifiable part, since these types may change from a process to another and during the evolution of the same process. Consequently the process interpreter cannot rely on or assume any specific information about process model types.
2. The database schema is to be changed at run-time. This means that it must be possible to apply changes to the schema concurrently with the execution of other applications. Obviously, suitable mechanisms have to be put in place in order to ensure the proper degree of synchronization and consistency.
3. Since the definition of types in the modifiable part may change, we have to support migration of the existing objects from the old definition to the new one. This may be done by providing some mapping between types, and/or supporting type versioning.

Requirements #1 and #3 can be further characterized with respect to the features that must be offered by a candidate OODB system. Each process

engine must be able to dynamically define and execute queries and update operations on the database. This means that the OODB system must provide a programmatic interface similar to what is offered by several relational databases. In particular, it must be possible to dynamically invoke the OODB language interpreter with a parameter representing the required operation. In SQL-based systems, for example, this feature is called *dynamic embedded SQL*. The same mechanism is needed to define and apply changes to the OODB schema.

These functionalities have to be complemented by basic transactional mechanisms to support the definition of atomic actions: they are needed to ensure that the different process engines access the database in a controlled and synchronized way.

3 Early experiences with the O_2 system

We have been conducting an experimentation to build the SPADE repository using the O_2 database [4]. O_2 is an object oriented database management system supporting multiple inheritance, polymorphism and static type checking.

The process engine executes the SLANG net specification by selecting at each execution cycle a transition to be fired. The firing of a transition involves the evaluation of its guard and the execution of its action. The guard and action specifications are loaded at run-time from the OODB and then interpreted.

O_2 provides a query language called $O_2 SQL$, offering a programmatic interface in which it is possible to require the execution of a query by passing it as a string to the SQL interpreter. In this way, we partially meet requirement #1, since we can define at run-time queries implementing transition guards, independently of the process model we are enacting. The same technique, with minor modifications, may be applied to execute the actions associated with transitions. Such modifications are due to the fact that actions involve the creation of new objects.

As far as requirement #2 is concerned, the current available versions of $O_2 SQL$ does not support dynamic schema manipulation. To overcome this problem one possible solution is to first generate the DDL (Data Definition Language) code, implementing the required modification, and then compile it, thus adding it to the application context. This operation could be done by a process working, in parallel, on the same context. A version of the $O_2 SQL$ support-

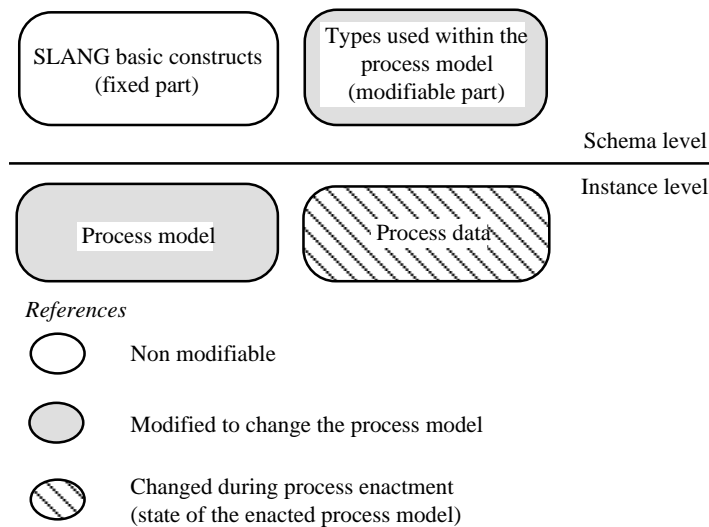


Figure 1: Structure of the SPADE repository.

ing run-time schema manipulation is scheduled to be released soon. This new feature will greatly facilitate the accomplishment of requirement #2.

Finally, considering requirement #3, O_2 does not provide any support to type migration (this will be included in a future version of O_2). The explicit conversion of all existing objects of a modified class may take a lot of time, making this operation unfeasible.

4 Concluding remarks

We have identified some requirements for the evolving SPADE repository and discussed our early experience with a commercial OODB system. Much research effort is still needed. In the near future our work will be centered on further evaluation of O_2 and of other available systems, such as DEC Object/DB and Gemstone.

This work is being carried out within and partially supported by the ESPRIT III project 6115 GOODSTEP (General Object-Oriented Database for the Software Process).

References

- [1] Sergio Bandinelli, Alfonso Fuggetta, Carlo Ghezzi, and Sergio Grigolli. Process Enactment in SPADE. In *Proceedings of the Second European Workshop on Software Process Technology*, Trondheim (Norway), September 1992. Springer-Verlag.
- [2] Sergio Bandinelli, Alfonso Fuggetta, and Sandro Grigolli. Process Modeling-in-the-large with SLANG. In *Proceedings of the 2nd International Conference on the Software Process*, Berlin (Germany), February 1993.
- [3] P.A. Bernstein. Database system support for software engineering—an extended abstract. In *Proceedings of the Ninth International Conference on Software Engineering*, pages 166–168. IEEE, 1987.
- [4] O. Deux. The O_2 System. *Communications of the ACM*, 34(10), October 1991.
- [5] Wolfgang Emmerich, Wilhelm Schäfer, and Jim Welsh. Suitable Databases For Process-centred Environments Do Not Yet Exist. In Jean-Claude Derniame, editor, *Proceedings of the Second European Workshop on Software Process Technology*, volume 635 of *LNCS*, pages 94–98, Trondheim (Norway), September 1992. Springer-Verlag.
- [6] M. H. Penedo and C. Shu. Acquiring experiences with the modelling and implementation of the project life-cycle process: the PMDB work. *Software Engineering Journal*, pages 259–273, September 1991.