

Mechanisms and policies for federated PSEEs: basic concepts and open issues

C. Basile, S. Calanna, E. Di Nitto, A. Fuggetta, M. Gemo

Politecnico di Milano - CEFRIEL

Via Emanuelli, 15

20126 Milano

{basile, calanna, dinitto, alfonso, gemo}@mailer.cefriel.it

Abstract. In the past years there has been a significant trend towards the establishment of distributed software development activities, where different organizations cooperate to develop a composite software product. This trend originates in the geographical distribution of many companies, quite often spread over different countries and even continents. Another motivation for this phenomenon relates to the increasing number of projects that are carried out by consortia or clusters of companies, each of them contributing with specific expertise to the final result. This scenario demands for improved software development technology that is able to support both the software development process at each local site, and the global interaction of the different organizations. In particular, process technology must be able to guarantee the autonomy of each site as far as the local software development process, practices, and artifacts are concerned. At the same time, it must be able to superimpose a coordinating process across all organizations to consistently drive the distributed development activity. The issue is critical for the effectiveness of process technology in an industrial context. In this paper we provide a discussion of the issue and propose a tentative research agenda for the next years.

1. Introduction

In the past years there has been a tremendous shift in the attitude and approach used in software development. Software is increasingly produced by organizations that are geographically distributed and that often encompass national and enterprise boundaries [1]. This internationalization and distribution of work is enabled by the availability of effective and low-cost telecommunication technologies and is caused by many factors:

1. The laboratories of most large companies are geographically dispersed, often across national and even continental boundaries. Often, the company need to jointly exploit expertise and skills that are not concentrated at the same site.
2. In most research and development programs (such as the European Union Framework Program and the European Space Agency contracts), a prerequisite for a project to be acceptable is that it is proposed and carried out by a consortium of different organizations, located in different countries.

3. Industrial joint-ventures for the development of advanced products often demand for the coordination of different development teams from different companies, each of them contributing to the final result with specific expertise and capabilities.

In general, these software development activities are carried by a set of independent and autonomous *organizations*. We use the term “organization” to identify an autonomous entity in charge of executing a specific process. The interaction among these organizations can be characterized by considering a series of factors:

1. *Geographical distribution*: different organizations may physically reside in the same building or may spread across continents. This factor has an influence on the kind of interactions that can be established among different organizations. For instance, depending on the distance among organizations, there might be a shift from “physical” meetings towards video-conferences and telephone conferences.
2. *Homogeneous vs. heterogeneous processes and practices*: the processes and practices are the same for all teams or may differ from team to team. This factor has a strong implication on the procedure used to exchange artifacts and to coordinate development activities.
3. *Homogeneous vs. heterogeneous technologies*: the software development and communication technologies being used are the same for all organizations or may differ from organization to organization. Again, this factor has a strong impact on the ability to effectively exchange information and coordinate among all organizations.
4. *Single company vs. multiple companies*: the different organizations involved in the development activities are from the same company or from different companies. This factor has a strong impact on problems such as to intellectual properties and exploitation of results.
5. *Single nation vs. multiple nations*: the development organizations are from the same country or from different countries. In this case, problems may arise as a consequence of the different laws and procedures adopted in different countries. Moreover, the different languages and cultures have to be harmonized and jointly supported across the different organizations.

This approach to software development greatly affects process support technologies, that must be able to provide assistance to such multi-site and, potentially, multi-company development activities. A first investigation on this issue has been undertaken by Ben-Shaul and Kaiser in [2]. They propose an environment, Oz, that results from the composition of different instances of PSEEs, each one devoted to support the development process executed by a single organization. All these PSEEs execute autonomously according to their own processes, but can interact to accomplish common activities such as the integration test of software components independently developed by different organizations. In Oz, the strategy through which a common activity is executed is called *summit*. In a summit, a site acts as the coordinator of the common activity. It receives from the other sites all the data

needed to execute the common activity, executes the activity, and sends the results back to the other sites. This behavior is obtained by directly implementing the summit protocol and procedures as a basic mechanism of the PSEE.

We argue that this approach, even if certainly important as a first and original contribution to address the problem, cannot be considered a general solution. In the next section, we provide some motivation for this position and outline our current research agenda.

2. The problem

We use the term *organization* to indicate an autonomous group of people who is operating as part of a *joint project*. The interaction among different organizations is usually occasional, as each of them maintains full control of its data and process. The set of all organizations cooperating in a joint project is called *federation*.

To start a joint project, the responsibilities of each organization are established and data to be exchanged are identified, along with an *inter-organization process*. Such a process specifies how organizations cooperate.

An inter-organization process is built by exploiting specific *inter-organization policies* that specify consistent and standard approaches used by organizations to interact. As discussed in the previous section, the summit approach supported by Oz is a possible inter-organization policy. Other examples might be the following:

- *Master/slave*: a set of organizations (slaves) export a set of services they offer to the federation (e.g., the inspection of a design document). These services are transparently invoked by an organization who plays the role of the master process. While in a summit the coordinator performs a local activity by issuing requests of data to other organizations, in this scenario the master organization issue service requests. Moreover, it does not know who is going to provide the service. So, in the summit approach the unit of collaboration is conceptually a set of information (i.e., the requested data), while in the master process approach the unit of collaboration is a remote service (i.e., a process fragment executed remotely) transparently accessed and executed.
- *Traveling ticket*: a request of service is represented by a ticket traveling in the federation. The organization that receives the ticket performs an operation depending on its own state and the state of the ticket. The ticket is then forwarded to other organizations that are dynamically determined according to the result of the local computation. This approach is used in many anomaly management processes.
- *Traveling agent*: an autonomous process travels across the federation. While the traveling ticket is a passive element, a traveling agent is a active entity that is able to autonomously perform a computation and determine its route across the federation.

In the summit and master/slave approaches, the inter-organization policy is embodied in the process executed by each organization locally. In the traveling ticket

and agent approaches, the central elements are the ticket and the agent, respectively. So, in the first approach the knowledge of the shared process is shared by all the organizations that manipulate the state of the ticket. In the last approach, this knowledge is embodied in the agent itself, that is, reasonably, generated by the organization that started the interaction.

Notice that these strategies are just examples of possible inter-organization policies. In general, there might be different variations of the same policy depending on specific needs and choices of the federation being observed. For instance, the master/slave policy can be enriched to take into account situations where different slaves can offer the same service, and some load balance policy is used to select the slave that will answer to a specific service request. Even more, new policies exploiting different paradigms may be identified. Notice also that an inter-organization process can exploit one or more inter-organization policies. For instance, in the same federation, an organization can offer an inspection service according to a master/slave approach, while other organizations may cooperate according to the summit policy.

Process support should guide both the definition and the enactment of inter-organization processes. Presently, no existing PSEE (except for Oz) provides the features needed to support such processes. In fact, PSEEs are conceptually centralized: namely, even when they support process distribution (typically, on different sites of a network), they enact a conceptually unique process model at a time. Moreover, most PSEEs rely on a logically centralized repository and, in general, do not provide, at the level of the process modeling language, suitable constructs to control the distribution and coordination of autonomous process model fragments. This contrasts with the idea that independent organizations may follow completely different processes in executing the activities they are responsible for. The execution of these processes is explicitly performed at the organizations' locations, and each organization has its own autonomy as far as the execution of its process and the access control to its data are concerned.

The solution that seems to take into account both the requirements of autonomy and interaction among organizations is to enable interaction among PSEEs by composing a *federated process-centered software engineering environment* (FPSEE). This way, each organization can maintain its own process support environment, enriched as far as interaction with the other organizations is concerned.

3. Directions for a solution

FPSEEs have to provide specific features both for the definition and the enactment of an inter-organization process, in order to take into account the peculiarities of such a process. As for definition of the process, the main issue is that the environment should allow several inter-organization policies to be implemented and combined. We argue that a solution to this issue is to provide a set of basic operations as the basic building blocks that make it possible the specification of any inter-organization policy. For instance, the operation "search for the physical location of the site X" can be used in any of the policies envisaged above. It can be combined with the operation

“request the execution of service X at physical location Y” to implement a master/slave policy. Clearly, basic operations should be powerful enough to make it simple the specification of any reasonable inter-organization policy.

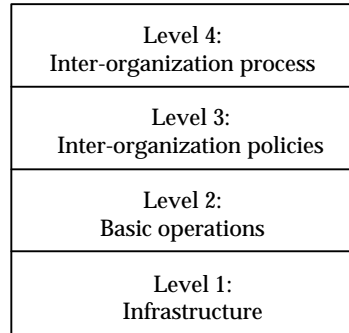


Fig. 1. Levels of abstractions in a federation.

The enactment of an inter-organization process requires that sites can be located over the network, that communication is enabled among sites, that data can be safely exchanged, that some access control policies are executed. All these requirements have to be fulfilled by an *infrastructure*, on top of which basic operations are implemented. Typical examples of infrastructure components are TCP/IP, FIELD-based message servers, DCE, and CORBA.

In summary, there are four levels of abstraction that can be used to characterize a federation. They are illustrated in Figure 1. FPSEEs must support the definition and the enactment of inter-organization processes located at Level 4. Thus, existing PSEEs and PMLs must be extended to support the creation of a FPSEE by implementing some of the lower level elements shown in Figure 1 and by providing an API to be used for implementing the higher levels. This can be achieved by exploiting any of the following three strategies:

- Inter-organization policies of Level 3 are directly implemented as constructs provided by the PML: the PSEE/PML covers Levels 1, 2, and 3. This is the approach adopted in Oz. Clearly, in our view this approach should not be followed, since it freezes a specific inter-organization policy in the PML.
- Basic operations of Level 2 are built on top of the PML/PSEE as specific process fragments: the PML/PSEE is part of the infrastructure, but it is not aware of the semantics provided by the basic operations. This is the approach we have followed in our experimentation.
- Elementary operations of Level 2 are built as new constructs of the PML: the PML/PSEE belongs to both the operations and infrastructure level. This is the solution we are aiming at on the long term, once our experimentation has produced proved results. This will be achieved by consolidating in new PML constructs the semantics of the operations that are now implemented through

specific process model fragment (as indicated in the previous point).

We are currently experimenting these ideas developing a FPSEE based on SPADE [3,4]. In particular, we have developed the virtual machine of Level 2, and we are currently developing specific process fragments implementing both Level 3 and Level 4. On the short term, we aim at demonstrating the usefulness and generality of such virtual machine of Level 2. On the long term, our goal is to exploit a standard system like CORBA to implement the infrastructure enabling the access to each site. This solution would enable the interaction among non-homogeneous PSEEs providing the same CORBA compliance.

References

- [1] C.W. Loftus et al. *Distributed Software Engineering*. Prentice Hall Publisher. 1995.
- [2] I.Z. Ben-Shaul and G.E. Kaiser. *A Paradigm for Decentralized Process Modeling*. Kluwer Academic Publisher. 1995.
- [3] S. Bandinelli, M. Braga, A. Fuggetta, and L. Lavazza. The Architecture of the SPADE-1 Process-Centered SEE. In *Proceedings of Third European Workshop on Software Process Technology*, Grenoble, France, February 1994.
- [4] S. Bandinelli, E. Di Nitto, and A. Fuggetta. Supporting Cooperation in Software Development. Technical Report 95-032, Politecnico di Milano, Piazza Leonardo da Vinci 32, I-20133 Milano, Italy, July 1995 (submitted for publication).