# SOLVING INTEROPERABILITY PROBLEMS ON A FEDERATION OF SOFTWARE PROCESS SYSTEMS

Mohamed Ahmed-Nacer
*Computer Science Department/ Faculty of Electrical Engineering, University of Science and Technology Houari Boumediene, BP 32, El-Alia 16123 Bab-Ezzouar Algeria*

Med-Amine Mostefai
*Software Laboratory, Center of Research in Scientifique and Technical Information 03 frère Aissou, Ben Aknoun, 16030 Algiers, Algeria*

Keywords: Software engineering, software process, interoperability, federation, development platform.

Abstract: Software process components that share information and that cooperate for common tasks lead to multiple problems of interoperability. Some based-interoperability approaches have been proposed. However, more problems remain to be solved to enable the heterogeneous process components interoperability at execution level. This paper presents a process-based approach (architecture) for the federation of software process systems. Based on this approach, we focus on its implementation problems for the process execution interoperability. We show how we solve these problems and we discuss their implementation through the main development techniques of distributed applications.

## 1 INTRODUCTION

The first Process Centered Software Engineering Environment (PSEE) approaches such as EPOS (Conradi, 1995), SPADE (Bandinelli, 1996), APEL V.3 (Dami, 1998), MARVEL (Kaiser, 1988) and OZ (Ben-Schaul, 1998) didn't met the requirements for a process support environment because of monolithic approaches (single formalism, non-openness, weak evolution...).

Many researchers have explored the problem of PSEE interoperability and multiple systems have been proposed through different approaches such as the multi-view approach (Sommerville, 1995) (interoperability at model level), distributed process/workflow approach (Bolcer, 1996) (heterogeneity at execution level) and federation of distributed process engines (Ben-Schaul, 1998). However, the federation of process components remains a real challenge with regard to the multiple problems of interoperability to solve, especially at process execution level.

In this paper, we present a new conceptual approach (architecture) of federated PSEE's and its associated implementation problems. We discuss how we solve these problems through the main development techniques of distributed applications: Corba (Miller, 1996), Dcom (Williams, 1994) , EJB (Sun, 1999) and Soap (W3C, 2000).

Section 2 presents the different approaches of the federation architectures and the associated problems for process components interoperability. Section 3 addresses these problems of interoperability, mainly at process execution level. Solutions for these problems are given and their implementations are discussed. Section 4 concludes this paper.

## 2 FEDERATION OF INTEROPERABLE PROCESS COMPONENTS

Many works have been addressed for the interoperability of process components (Cugola, 2000). Multiple approaches have been proposed



Ahmed-Nacer M. and Mostefai M. (2004).
SOLVING INTEROPERABILITY PROBLEMS ON A FEDERATION OF SOFTWARE PROCESS SYSTEMS.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 591-594
DOI: 10.5220/0002639805910594

such as the state-based approach (Estublier, 1999), (Heimbigner, 1992), the control-based approach (Orlafi 1997) and the process-based approach (Estublier, 1998).

This last approach seems adequate for software process support environments (the recovering problem is solved through the use of the common state**,** flexibility at execution level, control of the federation behaviour, ...). In this approach, two specific process components are introduced. The first one called "*Common Process Sensitive System (PSS)"* allows, through its associated model, to manage the global federation behaviour. The second process component, called *"PSS of interoperability",* plays the supervisor role.

However , its implementation causes multiple problems of interoperability at execution level. Next section presents these problems and tempts to propose a solution for each of them.

# 3 PROCESS-BASED INTEROPERABILITY: ARCHITECTURE, PROBLEMS & SOLUTIONS

We present a new conceptual approach (architecture) of federated PSEE's and its associated implementation problems. We discuss how we solve these implementation problems through the following development techniques of distributed applications **:** Corba, Dcom, EJB and Soap.

## 3.1 Federation Architecture

The architecture that we retain to federate interoperable process components is designed through three levels (Figure 1): 1) *the foundation level* to coordinate the execution of the different process components and to manage the synchronous (services) and asynchronous (events) communications, 2) *the middleware level* to ensures the transparency of the delivered messages and the referred services into the federation and 3) *the component level* that contains the multiple components participating in the support of the process. We have identified three main problems : Openess and heterogeneity of the federation, communication infrastructure and external tool integration.

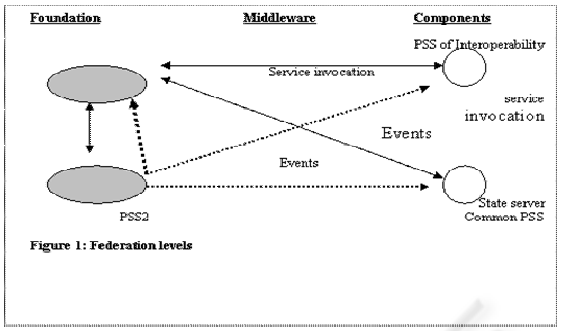


**Figure 1: Federation levels**

## 3.2 Openness and heterogeneity of the federation:

The heterogeneity of the process components (different description formalisms, different process engines,...) make somewhat difficult the desire to maintain the openness of the federation without modifying the federation model.

Concerning this problem, our approach is to associate, for each process component into the federation, an "abstract" parent that offers some services called "*minimal service*". This includes:

*Control operations*: They are required by the PSS of interoperability and concern the *launch* , the *end* and the *suspend* operations .

*Subscription operations*: They allow the state server to manage the global state of the federation.

*Message management operations* to be sent by the state server, the foundation level or the middleware.

*Import/export operations*: These operations are used to solve the problem of the different data formats.

The implementation of this "*minimal service*" ensures each process component to be added or replaced through the control operations (launch, end and suspend), to communicate via subscription and message management operations, and to duplicate easily the common state using the Import and export operations.

## 3.3 Communication infrastructure

In order to ensure the transparency of the synchronous and the asynchronous communications (event subscriptions and service communications), the middleware (that is in charge of this functionality) is decomposed through three components: 1) a message server component for message delivering to the components of the federation, 2) a subscription server component to

filter messages and to communicate the different messages sent by the state server to the concerned components and 3) a registration server to add new components (service storage) into the federation.

## 3.4 External tool integration

The needs to reuse existing tools and process components or to complete them with new ones, has naturally conducted us to search the way that gives the possibility to integrate these tools into the federation without major difficulties (minimal coasts).

As the external tools (to integrate) didn't provide the concept of "*minimal service*" (that we have introduced above concerning the problem of *openness and heterogeneity of the federation)* and in order to unifiy them with the other components of the federation, we associate to each external tool a special component called "proxy". This component is in charge to intercept the different requests from the federation and to translate them in interpretable commands to its associated external tools (in forms of command execution, API,...).

These "proxies" are provided with script interpreters allowing them to execute their associated external tool.

## 3.5 General discussion

We presented in the previous sections the major problems we can encounter when implementing federations and we gave our solutions for each problem. Being nothing more than a distributed application, we can implement our solutions using the techniques DCOM, CORBA, EJB or SOAP.

DCOM provides very interesting technologies such as Automation that can be an efficient method for external tools integration or GUID identification that allows a flexible identification mechanism. However, the main lack of DCOM is its Microsoft platform dependence.

CORBA provides an architecture for heterogeneous distributed applications and its services (as identification) can be a promising supports for implementing our solutions. However, its complexity can be a serious obstacle.

Analogically to DCOM suffering of platform-dependence, EJB lacks of its language dependence but remains a multiplatform and quite simple mechanism.

Based on popular standards XML and HTTP, SOAP is an astonishing simple solution for implementing distributed application. However, SOAP is still immature because it does not provide complete services and supports as the previous techniques.

The following tables summarize respectively the advantages and the drawbacks of each technique, the best techniques for each solution and the degree of quality of each technique to implement all the solutions.

| Table 1 | **Advantages** | **Drawbacks** |
|---|---|---|
| DCOM | Promising Technology<br>Oriented component | Microsoft platform-dependent |
| CORBA | Platform-independent<br>Language-independent | Complex system |
| EJB | Platform-independent<br>Oriented-component | Java language dependent |
| SOAP | Platform-independent<br>Language-independent | - Non-object structure<br>-Non-component structure<br>-No services |

| TABLE 2 | ***Recommended platforms for each solution*** |
|---|---|
| Openess & component heterogeneity | CORBA, SOAP |
| Communication infrastructure | CORBA, SOAP |
| tool integration | DCOM |
| Process component support | CORBA, DCOM, EJB |

| Table 3 | DCOM | CORBA | EJB | SOAP |
|---|---|---|---|---|
| Openess component heterogeneity | - | +++ | - | +++ |
| Communication infrastructure | + | +++ | ++ | +++ |
| Tool integration | +++ | ++ | ++ | + |

(+++) good (++) very acceptable
(+) acceptable (-) not recommended

In one hand, we notice that the use of CORBA with XML (import/export) can constitute an adequate solution for the process interoperability at

execution level. In the other hand, the use of a platform that derive some benefit from the flexibility of SOAP protocol and combined with the use of a programming oriented component concepts can also constitute a solution seeing the CORBA complexity.

DCOM and EJB can be used with multiple development tools. However, they are respectively limited to Microsoft systems and to Java language.

**F**or the process component support aspect that ensures the development of stable process components and that make them easy to maintain, we notice the ability of DCOM, CORBA and EJB to offer some supports for component development. DCOM allows the development oriented-component using the COM technology (the ActiveX components are a concrete exemple), CORBA offers the CCM technology (Corba Component Model) and EJB is provided with Java Beans technology.

# 4 CONCLUSION

The work presented in this paper deals with the problem of interoperability aspects of federated PSEE's (Process Centered Software Engineering Environments). Our first concern was to highlight the multiple implementation problems, and we have focussed our work on the mechanisms that enable the interoperability of heterogeneous process components at execution level

We have proposed solutions to solve interoperability problems on a federation of software process systems according to different aspects (openness, communication infrastructure, external tool integration and format heterogeneity). The implementation of these solutions has been studied through the well-known development techniques of distributed applications (DCOM, CORBA, EJB and SOAP).

The general discussion above (section 3.5) shows that there is a variety of solutions according to the process federation goals. However, all the solutions remain open and can be improved in respect with new protocols and standards (for instance, WSDL or .net platforms).

Our exploration has led us to conclude that a general infrastructure for interoperability is more important than a specific implementation. The current work aims at studying the concepts of a general architecture where interoperability is supported at modeling level and at a high level of abstraction (semantic interoperability), and enforced using heterogeneous and distributed process engines at execution level.

# REFERENCES


Bandinelli, S.,et al., 1996. Supporting cooperation in the SPADE-1 Environment" IEEE Trans. On Soft. Eng. Vol 22 , pp:841-865.

Ben-Shaul, I. Z. and Kaiser, G. E, 1998. Federating Process-Centered Environments: the Oz Experience. ASE Journal (Automated Softawre Engineering), Vol. 5, Issue 1, Kluwer Academic Publishers.

Bolcer, G. A. and Taylor, R. N.,1996. Endeavors: A process System Integration Infrastrucutre. 4th Int'l Conference on Software Process ICSP4.

Conradi,R. et al., 1995. PSEE architecture: EPOS process model and tools. Workshop on process-centered software engineering environment architecture. 20-23 March 1995. Milano- Italia.

Dami, S., et al., 1998. Apel: A Graphical yet executable formalism for process modeling. Automated Software Engineering 5(1):61-96 (1998).

Estublier, J., et al., 1999. Building a Federation of Process Support Systems. International Conference on Work Activities Coordination and Collaboration. San Francisco, California,USA. pp:197-206.

Estublier, J., and Barghouti, N. S., 1998. Interoperability and Distribution of Process-Sensitive Systems. Software Engineering for Parallel and Distributed Systems (PDSE'98). Kyoto.

Heimbigner, D., 1992. The Process Wall: A process state server approach to process programming. ACM/SIGSOFT –Conference on Software Development Environment. Washington, DC.

Kaiser, G. E, et al., 1988. Intelligence assistance for software development and maintenace. IEEE Software, 5(3).

Miller, J. A., et al., 1996. CORBA-Based Run Time Architecures for Workflow Management Systems. Journal of Database Management, Special Issue on Multidatabases, 7(1):16-27.

Orfali, R., et al., 1997. Client/server. International Thomson Publishing Company (F), 2sd Edition.

Sommerville, I., et al., 1995. Process Viewpoints . 4th European Workshop on Process Technology (EWST'95), Noordwijkerhout (NL), pp. 2-8.

*Sun Microsystems., 1999. Entreprise Java Beans ».* (www.sun.com).

Williams, S., and Kindel, C., 1994. The Component Object Model. A Technical Overview. Developer RelationsGroup Bibliothèque. MSDN (Microsoft).

*W3C., 2000. Simple Object Access Protocol.* (www.w3c.com).

G. Cugola, G., et al., 2000. Support for Software Federations : The PIE Platform. EWSPT: 38-53.