

Providing Highly automated and generic means for software deployment Process

Vincent Lestideau and Nouredine Belkhatir

Adele Team
LSR-IMAG , 220 rue de la chimie
Domaine Universitaire, BP 53
38041 Grenoble Cedex 9 France
{Vincent.Lestideau, Nouredine.Belkhatir}@imag.fr

Abstract. We present a new approach for the management and enactment of deployment process by a deployment processor ORYA (Open enviRonment to deplOY Applications). ORYA aims to integrate technology relevant to process support in deployment systems. The supposed context is a large scale networked organization. The deployment processor called ORYA provides deployment functionalities to distributed, autonomous, reactive processing entities representing workstations and servers. Based on these functionalities deployment processes can be enacted and monitored in an efficient manner. In this paper we focus on the distributed, multi-server, multi-client sites architecture of ORYA and the evaluation of its application to a real industrial case study dealing with a large transportation information system.

1 Motivation

Software deployment is a an ordered set of activities (software process) (Fig. 1), that will be carried out over time to deploy a software. It is generally described as a life cycle with encompasses several sub-processes as : Release, Install, Activate, Deactivate, De-install, De-release, Update and Adapt [3]. Deploying a software to a large scale networked user environment involves many activities like sites introspection and test for needed components, modification of various configuration files to establish the conditions needed by the new software, transfer...

During the last few years, software deployment has been the focus of intense activity in terms of products, standards and research work [16] for networked environments. The networked organizational structure of the companies implies new requirements on deployments systems. The current generation of deployment systems placed more emphasis on automation of process deployment in decent rally and distributed large installations of servers and stand-alone workstations.

In this paper, we propose a novel approach to build flexible, customizable and enactable software deployment systems based on the concept of executable descriptions. These objectives reduce considerably the cost of large scale deployment

setting. The key point here is that the deployment domain can benefit from process technology. The separation of process description from the monolithic tools would result in greater visibility of the process and greater flexibility. In this paper we consider this new approach in the light of a real industrial experience we carried out in software deployment for a large and complex transportation application. This paper gives an overview of this experimentation and goes on describing ORYA.

Section 2 presents the overview of the deployment processor ORYA. Section 3 gives more insight of the deployment process model. In section 4 we present the experimentation with an industrial case study in a transportation information system. Before the conclusion, the section 5 presents a first preliminary evaluation of our experience and some related works.

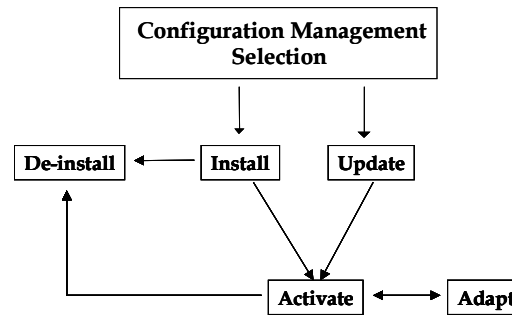


Fig. 1. The deployment life cycle

2 ORYA overview

Installing an application on one site can be seen as a simplistic manual activity, however if it is necessary to add some configuration requirements (like installing the more compatible version of the application) the activity becomes more complex. The installation remains always realizable manually from the target installation site or from a distant one. If moreover, we want to install the same application on many sites at the same time (but under different versions according to the installation sites), the installations become impossible to be performed manually. For this reason we have created a deployment PSEE [2] to completely automate the deployment process. In the following sections we describe our deployment environment and followed by the models used to allow the automation.

2.1 Architecture and Design of ORYA

2.1.1 Topology

For automation purposes, we have introduced three entities allowing to deploy applications on one or more sites; an entity to store the applications, another to represent the sites and a last one to manage the deployment:

- **Application Server (AS)** : It is a repository of packaged applications. A package is composed of a set of typed resources (data, scripts...) representing the application, a set of executable files allowing to install the application and a manifest describing the application in terms of dependencies, constraints and features (name, version...)
- **Target (T)**: It is the deployment target, where the applications have to be installed and executed. It can be a computer or a device (a stamping ticket for instance). Each target is described by a site model in terms of already deployed applications and physical description (disk space, memory...). This description permits to deploy and to install the application in the more consistent way according to the site configuration. Many user profiles can be taken into account.
- **Deployment Server (DS)**: It is the core of our deployment environment and is responsible to manage deployment. The DS searches the appropriate package to be deployed on the target, performs the package transfer (even through out firewalls) and guaranties that the new installed application works correctly. The DS has to deal with several problems such as dependencies or shared components, and must ensure that other existing applications continue to work.

2.1.2 The environment

Fig. 2 presents the environment in which we have experimented our deployment process.

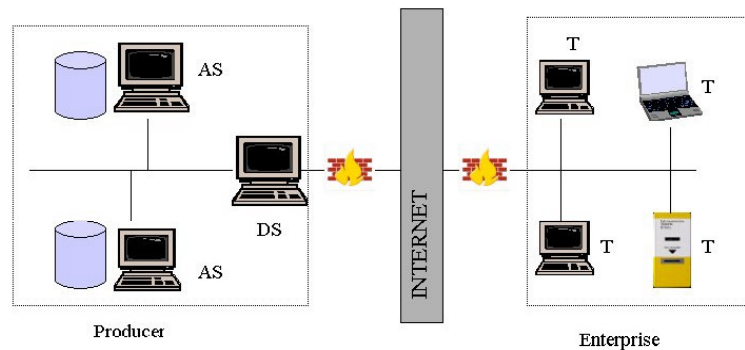


Fig. 2. Deployment environment

The producer hosts a set of application servers and one deployment server. It is connected to Internet via a potential firewall. The deployment targets belong to an

enterprise. The enterprise can be also behind a firewall. The goal of this experiment is to automate (via Internet) the deployment towards the enterprise.

2.2 Deployment process : models and concepts

To automate the deployment, it is necessary to define clearly many models for the application to be deployed, the deployment process, the targets, the application servers, etc. In this paper only the two first models are considered: the deployment process model (based on a process model) and the application model.

2.2.1 Deployment Process Model

Fig. 3 describes an implementation of a process model [13] related to deployment. An activity can be either a basic activity or a complete deployment process. The transfer or the unpackage activities are examples of basic activities. A deployment process is a composite activity, made by some ordered basic activities and deployment processes. Thanks to this generic model it is possible to describe in particular the activities (install, update, uninstall...) of the deployment life cycle (see the Fig. 1).

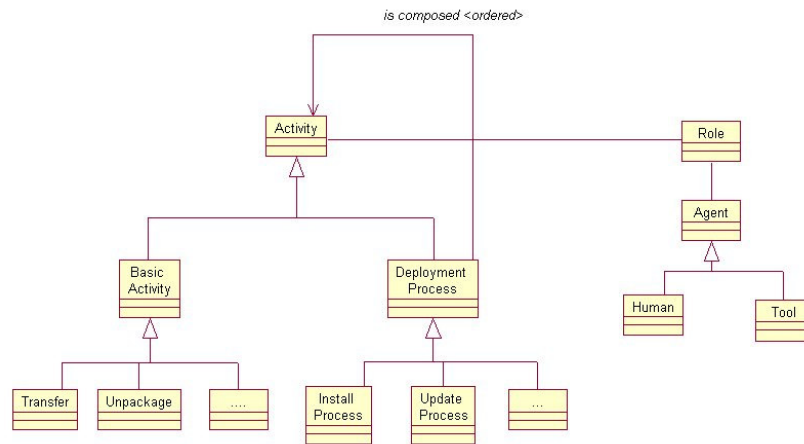


Fig. 3. Our deployment process Model

Each activity is played by a role executed by an agent. An agent can be either a human-user or a tool. In our case, we try to use often tools to automate as maximal as possible the deployment process. However in some situations a human-user is needed; for instance to allow an administrator to validate the deployment.

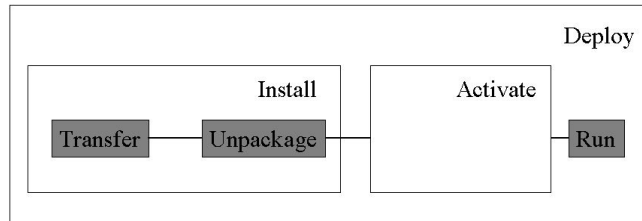


Fig. 4. A simple example of our deployment process

Fig. 4 is a simple example of a such deployment process model. A basic activity is represented by a gray box and a deployment process by a white box. In this example, we describe an activity (Deploy) composed of two deployment processes (Install and Activate) and a basic activity (Run). The Install activity is also composed of two basic activities (Transfer and Unpackage).

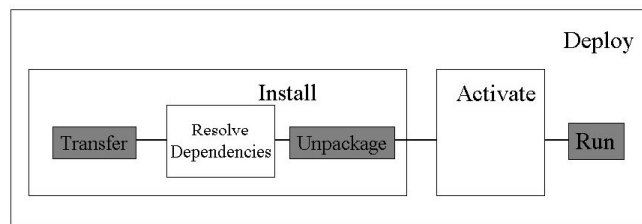


Fig. 5. A more complex example of our deployment process

The same activity can be more complex (Fig. 5). For instance, the Install activity can be composed of two basic activities and a deployment process (Resolve Dependencies), whose function is to resolve the dependencies of the application. Section 3 shows a practical example we have tested in an industrial environment.

2.2.2 Application Model

This application model (Fig. 6) describes an application from the deployment point of view and more particularly in the case of installation. Consequently our model does not contain some information (like connection between components) generally present in other application models like CCM [11].

Many information are necessary:

- Information on the application itself (specified in a manifest) like the application name, the producer name, the version number, the license number...
- Information on resources composing the application (code, components...) like the component version, its location...
- Deployment constraints. Two types of constraint can be distinguished: software and hardware constraints. An application “A” that can not be deployed if another application “B” is already installed is an example of a software constraint. A

affected. The process must then resolve the dependencies, transfers and finally installs the application.

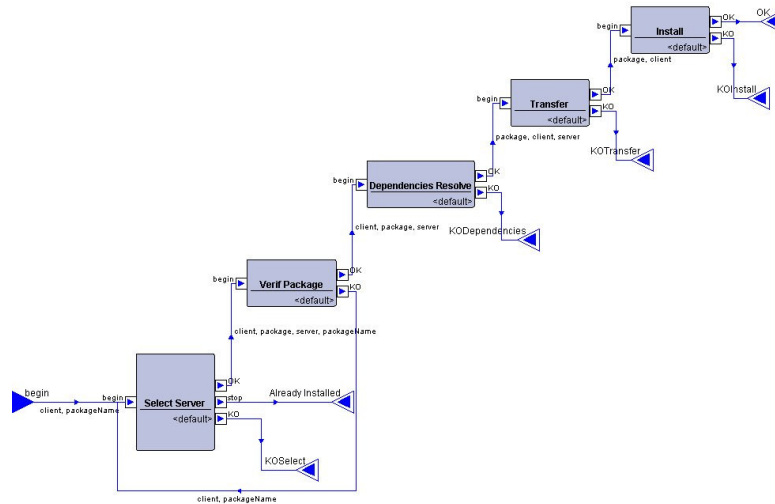


Fig. 7. The deployment process

The presented process is composed of many deployment processes (install...) and many basic activities (transfer...). These activities will be described in more details in the following sections.

3.1.1 Select Server Activity

This basic activity () receives in entry the target and the application names. Its goal is to prepare the deployment. First it verifies that the application is not already installed (otherwise the deployment process is stopped (stop port in the figure)). Then the deployment server looks for an application server containing one of packages representing the application. Once an application server is found it must verify that the package can be deployed on the target (the activity is terminated by the OK port). If the package cannot be deployed another package will be searched. If no package is found (or if no package can be deployed) the activity is stopped (terminated by the KO port) and the deployment process is finished.

3.1.2 Verif package Activity

This basic activity (Fig. 7) receives in entry the package, the target, the application and the application server names. Its goal is to verify that the package can be deployed on the target. These verifications concern more particularly hardware constraints like memory or disk space. We distinguish software constraints and software dependencies. Unlike a software dependency resolved during the next activity, a software constraint must be verified before the installation. All software and hardware constraints are specified in a special file of the package: the application descriptor. If at least one of these verifications is not satisfied, the activity is

interrupted (terminated by the KO port). In this case, the process go back to the previous activity to look for another package. In the other case the activity is successfully finished (terminated by the OK port) and the process continues to resolve the dependencies of the package thanks to the next activity.

3.1.3 Dependencies Resolve Activity

It receives in entry the result of the previous activity: the target and the application server names and the package and checks the software dependencies. A dependency must be resolved during the deployment (in case of strong constraint) or after the deployment (in case of light constraint). For each dependency a checking is done to verify if it is not already installed on the target. If the dependency is not installed the deployment server executes a new deployment process to deploy the dependency on the target. If problems occur during the process related to the dependency, many strategies tell if the initial process must be stopped or not. If all dependencies are resolved, the activity is successfully finished and the transfer activity can be started.

3.1.4 Transfer Activity

It realizes the transfer of the package from the application server to the target. In reality the transfer can be achieved in two steps: first the deployment server retrieves the package from the application server and then it transfers it to the target. If this activity fails (network breakdown for instance), the deployment can be stopped or delayed. The install activity can started if the transfer activity is successfully terminated.

3.1.5 Install Activity

It is a complex process and is responsible of the physical installation. In our vision an installation is composed of an extract activity of the package, a set of installation steps (“Install-Step”) and a set of uninstallation steps (“Uninstall-Step”). A step is composed of a script to be executed and some associated verifications. If a problem occur during the installation the process must be able to return to the initial state. Therefore the install activity is also composed by some uninstall steps. The uninstall is not a basic activity, because potential problems may occur at different moments of the deployment and the uninstall process can be different according to this moment. The next paragraph is dedicated to the install process.

3.2 Focusing on the install process

As seen before, the install process (Fig. 8) is one activity of the general deployment process (Fig. 7). To our point of view an installation can be decomposed in several steps. This decomposition responds to the need of carrying out checks during the deployment. Each checking (or checking set) implies the creation of a step in the installation. It is supposed that a step corresponds to the execution of a script followed by a checking associated to the result of this script. All these information are describing in an XML Schema file (named “manifest”) containing in the package. A complete description of this XML schema [18] is beyond the scope of this paper. The

package contains also data corresponding to the application. Now, we describe the different activities belonging the install activity.

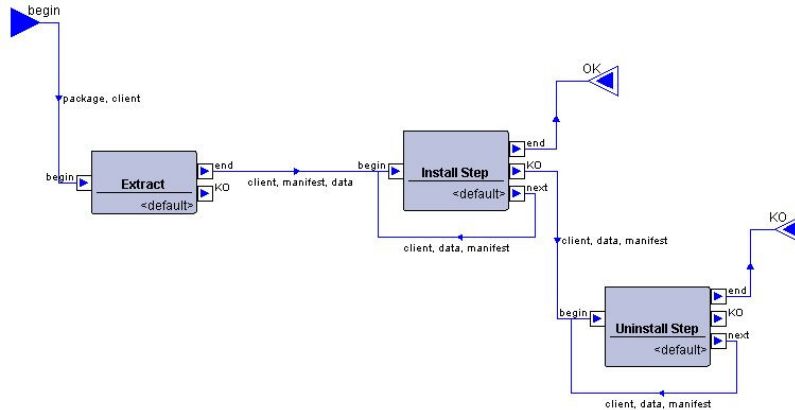


Fig. 8. The install process

3.2.1 Extract Activity

It extracts the package on the target in a temporary folder. Once extracted the manifest informs the process how many install/uninstall steps composed the install process and how these steps can be performed. The package contains also some resources, scripts and checking (designated as “data” in the process).

3.2.2 Install Step Activity

Each install step activity is composed of two basic activities (Fig. 9): execution and checking. The former executes the script related to the install step and the latter performs the necessary verifications.

If at least one step does not succeed, the install process must be interrupted. In this case the process must undo all the modifications already done; this is the responsibility of the uninstallation process. In case of success of all install steps, the installation is successfully finished as well as the deployment process.

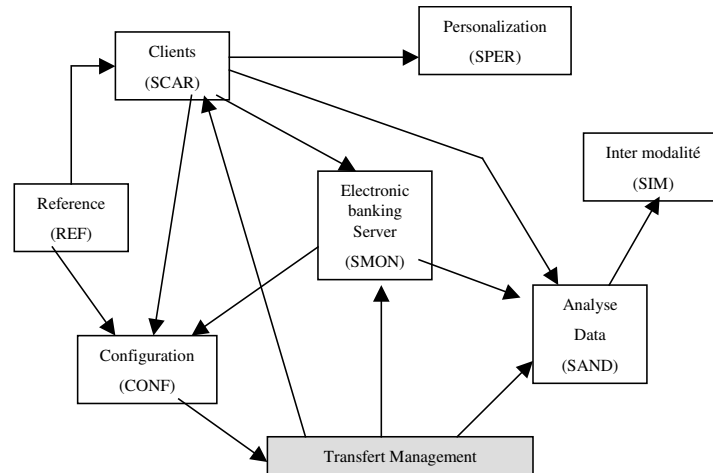


Fig. 10. Architecture of Centr'Actoll Application

Fig. 10 shows the general application of Centr'Actoll. This application exists in different versions (according to the number of servers used). These servers may have some dependencies between them. These dependencies are represented in Fig. 10 by the arrows. It exists other dependencies for a server than the other servers. For instance the SCAR depends of ORACLE and TOMCAT software. And each server has some constraints like the space-disk size (of 1Go to 20Go), the OS (Windows), X25 network cards, etc.

From this description of the architecture, we have created the package of each server with its description and we have tested our deployment environment for this application in the real context of the Actoll enterprise. In addition to deploy the application, we have identified some deployment requirements (see the next section) and realized an implementation of our environment trying to respect all of these requirements (distribution, monitoring...). We have used two technologies or tools developed in our team:

- A process engine with its graphic process creation tool named Apel.
- The federation technology allowing to execute and to control in a distant way a tool even over firewalls [5] [10].

4.2 Requirements

We have identified several requirements such as:

- **Configuration:** The application can be installed according to different configurations; for instance one machine by sub-application or several sub-applications by machine.

- **Distribution:** Deployment must be possible either via Internet (even in case of firewalls) or via a local network (in particular for some test and developing reasons).
- **Automation level:** The deployment system must be able to offer different (and customizable) levels of automation. For instance, deployment can be completely automated or contrarily each deployment step must be validated.
- **Security level:** Generally, client company doesn't wish that other companies install some applications on their network. In this case, our system must be able to interact with administrators to validate the wished (by producer company, i.e. Actoll). This requirement is strongly associated with automation requirements (see above).
- **Scheduling & Performance:** In case of large scale deployment, a deployment system must be able to handle different strategies like big-bang (application is deployed on all the target machines), or incremental (application is deployed on one (or on group of) machines). Network characteristics (like bandwidth...) must be also taken into account during the deployment.
- **Monitoring:** We must be able to monitor the deployment.
- **Roll Back:** If the deployment doesn't work for a machine, we must inform the system and/or administrators and we must undo all deployment activities already realized to come back to the old configuration.
- **Tests and Verifications:** In case of the deployment of an application, some verifications must be realized at the end but also during the deployment. This verifications affect the next deployment steps.

5 Evaluation of the experience

Many ad-hoc solutions and tools for deployment have been proposed providing monolithic and hard coded solutions. Current deployment systems do not support the explicit definition of the deployment process independently from the tools and do not let to customize features like adding specific deployment tools. Deployment facilities are either not powerful enough and cannot be sufficiently tailored to application needs or they provide too much functionality which in most cases remains unexploited because unsuitable but increases the system cost and affect the system performance.

In several industrial organizations, the complexity of deployment is becoming such that more systematic approaches are required as the limitations of ad-hoc approaches experienced daily. More advanced requirements arise and request the customization of the system to the evolving and specific application needs.

For example, in our case study, an application implementing the Actoll process requires the basic PSEE functionality and the ability to tailor the process and integrate specific deployment tools. Administrator can customize each of the specific tools and integrate them in the core. For example they can introduce a new introspection tool specific to a new site configuration. Likewise, they can have the process uses a

specific packaging tool. This requires process functionality which integrates with deployment tools executed in a networked environment.

This characteristic is identified as one of the requirements for a new generation of deployments system architecture.

These reasons have led to the ORYA approach. ORYA is a highly customizable deployment processor which can be enhanced with specific tools and process. ORYA has the ability to be changed and configured for use in a particular context.

We have presented a solution outline for making large scale deployment. This approach leads processor make it possible to control the way tools interact with the engine, as such making it possible to ensure consistency.

Finally the economic of our approach has been studied. Particularly for complex applications, the advantages that can be gained using a deployment processor are high. The overhead for deploying applications has been reduced considerably from one week (manually) to one hour to deploy a server using the automation deployment processor ORYA. The productivity and quality of service of the system manager was increased because more applications are available with a better level of consistency. On this way, the system manager can devote more time to conceptual activities.

In summary the case study discussed in this paper has proven that ORYA can implement deployment process with different requirements. Additionally, the quantitative evaluation in terms of design and run time costs shows that the automation provides a viable alternative for application administrators who need customizable deployment functionality.

5.1 Related works

Deployment has been an issue for years. Many solutions have been proposed and implemented. Many existing industrial deployment tools exist like Tivoli [17] or InstallShield [15]. Often they are bound to one deployment process and they offer a unique deployment solution (to one target) or a multi-site deployment but with ad-hoc solutions. However these existing tools are often very efficient in some case (install process, configuration process...) but they do not cover all the deployment life cycle. ORYA allows to integrate these tools and to benefit of their capacities. There are research efforts to provide new technologies [1] [9] to integrate support for the whole deployment life cycle and addressing complex issues as dynamic adaptation for example [7] [8]. Some standardization works like [12] deal with some specific application models (component based models). None of these approaches provide a suitable software installation and common system for several companies. Software Dock [6] is an example that deploys generic applications to one target with a hard coded deployment process. Software dock has been developed in the University of Colorado. The aim of software dock is to provide a general solution for the

deployment life cycle. It allows describing, using the DSD (Deployable Software Description) formalism, a number of valid configurations.

Software Dock

ORYA shares several characteristics with Software Dock. They both propose new deployment architecture covering the entire deployment life cycle. However many features set ORYA apart from Software Dock.

- A key feature of ORYA design stems from the fact that our approach is more generic. For instance we offer the possibility to the administrator to describe the install process in terms of executions, strategies and verifications. However in software dock the process and deployment strategies are hard coded and cannot be modified.
- Another main issue is the integration into the deployment process of tools (legacy) which encapsulate different concerns . In this context, we provide a framework for the development of interfaces between the process system and applications (legacy tools). On this way; we separate process logic from activity logic which is embedded in user tools. This feature is not present in S/W Dock.
- A federated process component in ORYA that provides the support required to transparently distribute process execution across the enterprise. This feature is not present in S/W Dock.
- The coarse granularity level of the deployment process in software dock is the activity (install, update, adapt...) of deployment life cycle whereas in our approach the level is more fine.
- ORYA relies on a PSEE that provides process functionalities whereas software dock relies on a agent based platform.

Finally in contrast with software dock, ORYA aims at providing a generic and flexible architecture.

6 Conclusion

The paper presents a new deployment platform based process deployment engine which monitors the global architecture of a system and manages and executes the provided process model. Our approach is characterized by some important features :

- An open environment offering mechanisms to integrate legacy deployment tools and to choice dynamically the most adapted tools during the deployment.
- A process sensitive based system offering support to define, execute and monitor deployment process.
- A general purpose application model shared across the network. Could be instantiated for specific application models (for instance a component based model).
- ORYA puts an emphasis on large scale deployment (deploying applications on many targets sites).

- The design of ORYA takes multi-platform capability into account, and it is successfully being used in a mixed environment for instance Windows / Linux / MacOS.

In this paper we have described our generic platform and we have applied and implemented it on a real industrial case study: a transportation information system. We have shown by some scenarios the different facets of the deployment process. This research extends applicability of process systems since they haven't been applied to the deployment. We propose a generic deployment process support kernel to provide support for process management and execution. The first application of ORYA generates a positive feedback from the software administrator, responsible of the deployment tasks.

The contribution of this paper can be summarized as follows:

- It describes a novel architecture for a distributed environment providing the functionality for deployment process execution and monitoring.
- It presents and evaluates the results of a real case study

Further work remains necessary to cover the whole life cycle of deployment.

References

1. N.Belkhatir, P.Y. Cunin, V. Lestideau and H. Sali. An OO framework for configuration of deployable large component based Software Products. OOPSLA 2001. Tempa, Florida, USA. 14-18 October 2001.
2. N. Belkhatir, J. Estublier, and W. Melo. ADELE-TEMPO: An Environment to Support Process Modelling and Enaction. book Software Process Modelling and Technology, pages 187–217. John Willey and Son inc, Research Study Press, Tauton Somerset, England, 1994.
3. A.Carzaniga, A. Fuggetta, R.S. Hall, A. van der Hoek, D. Heimbigner, A.L. Wolf. A Characterization Framework for Software Deployment Technologies. Technical Report CU-CS-857-98, Dept. of Computer Science, University of Colorado, April 1998.
4. S. Dami, J. Estublier and M. Amieur. APEL: A Graphical yet Executable Formalism for Process Modeling. Kluwer Academic Publishers, Boston. Process Technology. Edited by E. Di Nitto and A. Fuggetta..Pages 61 - 97. January 1998.
5. J. Estublier, A. T. Le and J. Villalobos. Tool Adoption Issues in a Very Large Software Company-ACSE 2003, Portland, Oregon, USA, May 2003.
6. R.S. Hall, D.M. Heimbigner, A. van der Hoek, and A.L. Wolf. An Architecture for Post Development Configuration Management in a wide-area network. In Proceedings of the 1997 International Conference on Distributed Computing Systems, pages 269---278. IEEE Computer Society, May 1997
7. M. Ketfi, N. Belkhatir and P.Y. Cunin. Automatic Adaptation of Component-based Software Issues and Experiences. PDPTA'02, Las Vegas, Nevada, USA, June 2002
8. M. Ketfi, N. Belkhatir and P.Y. Cunin. Dynamic updating of component-based applications. SERP'02, Las Vegas, Nevada, USA, June 2002
9. V. Lestideau, N. Belkhatir and P.Y. Cunin. Towards automated software component configuration and deployment, PDTSD'02, Orlando, Florida, USA, July 2002.
10. A.T. Le, J. Estublier and J. Villalobos. Multi-Level Composition for Software Federations. SC'2003, Warsaw, Poland, April 2003

16 Vincent Lestideau and Noureddine Belkhatir

11. J. Mischkinisky - *CORBA 3.0 New Components Chapters* – CCMFTF Draft ptc/99-10-04 --
OMG TC Document ptc/99-10-04 October 29, 1999
12. Specification for Deployment and Configuration of Component-based Distributed
Applications. Proposal to the OMG MARS RFP: Deployment and Configuration of
Component-based Distributed Applications. <http://www.omg.org/cgi-bin/doc?mars/2003-03-04>
13. The Workflow Management Coalition Specification- The Workflow Reference Model,
january 1995, <http://www.wfmc.org/standards/standards.htm>
14. Actoll Enterprise Web Site: <http://www.actoll.com/en/presentation.htm>
15. Web Site InstallShield : <http://www.installshield.com/default.asp>
16. Object Management Group. Web Site : <http://www.omg.org>
17. Web site Tivoli, <http://www.tivoli.com>
18. XML Schema Web Site : <http://www.w3.org/XML/Schema>