SOFTWARE DISTRIBUTION ENVIRONMENTS FOR WORKFLOW MANAGEMENT SYSTEMS -THE CASE OF MQ SERIES WORKFLOW

Rainer Anzböck[1], Schahram Dustdar[2], Harald Gall[3]

[1] D.A.T.A. Corporation, Invalidenstrasse 5-7/10, A-1030 Wien, Austria
ar@data.at
[2, 3] Distributed Systems Group, Vienna University of Technology, Argentinierstrasse 8/184-1, A-1040 Wien, Austria
{dustdar,gall}@infosys.tuwien.ac.at

Abstract

Workflow Management Systems (WfMS) are increasingly gaining importance as a main building block for intra- and inter-organizational information systems, Enterprise Application Integration (EAI), as well as for Business-to-Business (B2B) systems on the Internet. WfMS are complex and require Software Distribution Environments (SDEs) for managing configuration, distribution, and deployment of their components. However, few research papers can be found in scientific literature that deal with SDEs for WfMS. This paper discusses architectural issues of SDEs in this context and presents a prototypical SDE-architecture for MQ Series Workflow, a Workflow Management System.

Keywords

Software Distribution Environment, Workflow Management Systems, Software architecture

1. Introduction

Workflow Management Systems (WfMS) are increasingly gaining importance as a main building block for intra- and inter-organizational information systems, Enterprise Application Integration (EAI), as well as for Business-to-Business (B2B) systems on the Internet. WfMS are complex and require Software Distribution Environments (SDEs) for managing configuration, distribution, and deployment of their components. However, very few papers can be found in scientific literature that deals with SDEs for WfMS. This paper discusses architectural issues of SDEs in this context and presents a prototypical SDE-architecture for MQ Series Workflow [17], a Workflow Management System. Software configuration, distribution, and deployment of enterprise information systems have changed dramatically through the Internet evolution from simple e-mail distribution of software to sophisticated distribution and configuration portals for software. Many commercial tools and environments have been developed and are in use in such portals: System Management Server [8], Marimba Management Solution [6], NETDeploy [4,5], WebStart [14], InstallShield [3], Tivoli Software Distribution [15], Rational ContentStudio [10] and many others [1]. We have performed a detailed evaluation of 12 products in the area of software distribution environments (SDEs) that defines the basis for our architectural considerations. Details on this evaluation can be found in [1]. Many concepts covering software distribution are also related to configuration management (CM) [2]. Producers require software configuration management (SCM) to be integrated in their development environments. Software distribution should be based on SCM information to ship or offer particular releases (or configurations) to customers. For that, many SDEs also cover some software configuration tasks such as management of development artifacts, product and release management, software description (languages), or software packaging.

The contribution of this paper is to discuss a software architecture for SDEs suitable for Workflow Management Systems (WfMS). This is achieved by discussing architectural properties, common components, and relationships across particular tools and products, as well as considerations of quality attributes of a reference architecture for SDEs derived from our previous analysis [1]. Our results are based on the aforementioned product evaluation and three case studies that functioned as means to distil architectural elements.

The paper is organized as follows: the next section provides a brief overview of Workflow systems. Section 2 describes a case study for Workflow systems and discusses architectural issues and requirements for SDEs. Logical- and process views for configuration, distribution, and deployment are presented in Section 3. Finally Section 4 concludes the paper.

1.1 Workflow Management Systems

Business processes in general and associated workflows in particular exist as logical models. A business process consists of a sequence of activities and activities are distinct process steps and may be performed either by a human or by a software system. A WfMS [18] enacts the real world business process for each process instance. Examples of business processes are purchase orders, insurance claims in administrative or production workflows or any software development process in a collaborative workflow. Any activity may consist of one or more tasks. Examples of tasks include updating a document, a database, or calling a customer by telephone. A set of tasks to be worked on by a user (human participant or software application) is called work list. The work list itself is managed by the WfMS. The WfMC [18] calls the individual task on the work list a work item. Workflow management systems have been defined as "technology based systems that define, manage, and execute workflow processes through the execution of software whose order of execution is driven by a computer representation of the workflow process logic" [18]. Workflow systems generally aim at helping organizations' team members to communicate, coordinate and collaborate effectively and efficiently. Therefore WfMS possess temporal aspects such as activity sequencing, deadlines, routing conditions, and schedules. WfMS are typically "organizationally aware" because they contain an explicit representation of organizational processes (process model). Traditional WfMS therefore present a work environment consisting of *roles* and their associated *activities and applications.* The goal of WfMS is to allow an organization to automate its business processes by instantiating them using a WfMS. More recently WfMS technology is utilized in Business-to-Business (B2B) systems, Enterprise Application Integration efforts and nearly in all large and distributed information systems.

Increasingly WfMS are found in software engineering processes. However, it is interesting to note that very few papers can be found in the scientific literature dealing with configuration, distribution, and deployment issues of WfMS.

2. Case study "Workflow"

The case study "Workflow" covers the configuration and distribution of a workflow system that is implemented using the MQ Series workflow client and server components. The system consists of a hierarchical model. On top, the domain concept separates multiple physical sites but shares a common workflow model. The MQ Series Workflow queue management provides synchronization. The next level consists of system groups that refer to a single location where one or more workflow systems are implemented. Systems grouped together use the same database and their communication is optimized. A single system consists of a one, two or three-tier architecture with a client and Web-server tier, a workflow server tier and a database server tier. Clients implement the workflow client API [18] and can be Web-browser, DCOM [7], DotNet [9], or Java-based applications using HTTP or RMI [12, 13] over IIOP [13]. Their communication can be consolidated by client concentrators that forward their requests. Depending on the protocol a Web-server with servlets or the EJB-based WebSphere [16] application server is used. The workflow tier has several functionalities that can be distributed over multiple servers. The database can be run in a cluster to provide load-balancing [16].

The model described here covers the distribution of a MQ Series Workflow system and takes into account multiple locations, multi-tier systems, distributed server architecture, a large number of clients and the use of client concentrators. The Configuration Management tasks are mainly covered by the workflow administrator and can be substituted or extended with a custom built implementation; however the model fits both scenarios. One benefit is the homogenous structure of distributing a single system; on the other hand the heterogeneous operating systems infrastructure requires advanced setup techniques. The workflow system provides setups for all server and client components.

Figure 1 provides a sequence of the key interaction of the components necessary to distribute a MQ Series workflow system. For distribution purposes, information about specific workflow system architectures has to be stored and maintained. A configuration database has to cover all installation-related domain, system group, and system node information. Clients in this case study are likely to be catalogued with an inventory system. Therefore an inventory database or a directory service is part of the configuration. A mechanism has to be provided that supports transfer of installation packages by implementing parts of the workflow API and querying the additional information. There exists an order dependency for installing workflow components. The server infrastructure is built before the client. The hierarchy has to be set up from a domain down to a system. An initial queue manager has to be installed for communication between components [17].

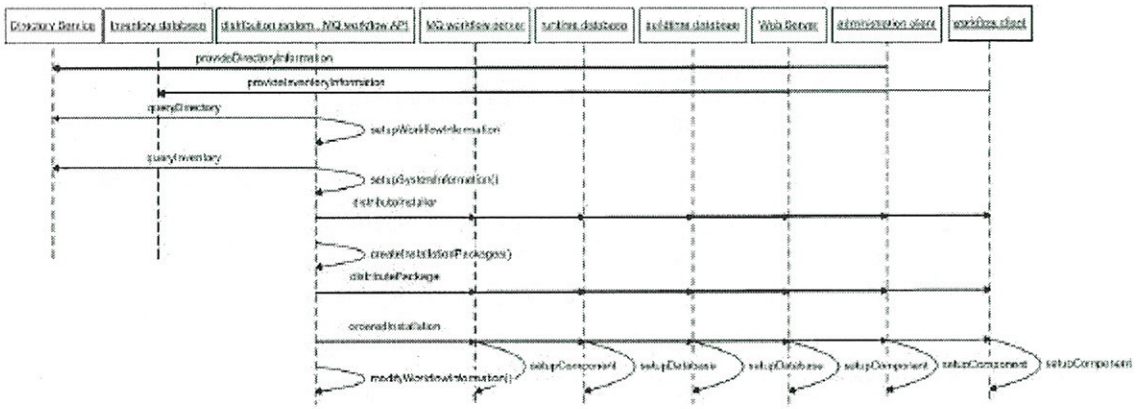

Figure 1: Case study "Workflow" processes

Next, the installation process, also shown in Figure 1, is described. All hosts except the workflow clients provide information to the directory service. Key information provided here covers system name, version and location, user and user group constellations and security policies. Additionally the installation process can be supported by storing which MQ Series Workflow server components and database systems have to be installed. An inventory like the Tivoli inventory module collects information about all hosts of a MQ Series workflow system. An inventory client provides system information that can be used during package creation for version selection and installation customization. The information collection process is done independently of the installation process. The distribution system initially stores dependencies between workflow components to enable ordered installation activities. Next, it queries the directory service to setup workflow information. During this operation a plan of installation activities is created considering the component dependencies. Each activity is related to a target host and a MQ Series workflow service to be installed. Furthermore, the inventory system is queried to select the product versions that fit the operating system and language requirements. Additionally the inventory provides information about installer services and workflow components, which are already setup. Based on the product and system description, the workflow information is setup, the activities are specified to whether install or upgrade installer services and workflow components and the packages that should be used are defined. From this point the distribution system is enabled to setup the installer components on the target systems. For a more complex distribution system like Tivoli additional distribution services have to be provided. Therefore a customized installation procedure and additional target systems have to be considered. This setup proceeds until all hosts are enabled to receive installer packages. During this process all required installation packages are created by importing or creating them with an external packager. After the distribution infrastructure is setup and all packages are ready for installation, the distribution activities from the installation plan are executed. First, except for deletions, the packages are distributed. An internal mechanism can be used or it can be delegated to a transport mechanism. Retransmission and resume operations have to be considered if unreliable transport is used. The state of the distribution has to be maintained for all activities to start the installation activities themselves. Depending on the distribution system, the system or the user decides whether to install, upgrade or reinstall the package. For example the Tivoli integration toolkit can be used to distribute components and to maintain state of the installation process. A manual installation is not recommended for workflow components. The activities are executed in order based on the initial dependency information and the workflow system is finally distributed consistently. From the products we described in [1] the Tivoli [15] distribution and Inventorying system meets most requirements for complex distribution tasks. SMS [8], Marimba [6] and others [1] shouldn't be taken into account as long as they are not implemented for other purposes. InstallShield [3] can be used as an external packager and version control systems can be customized to provide the required information.

3. Software Architecture

In this section a reference architecture is elaborated, based on the requirements and functionalities presented previously. The case study is discussed and conclusions for a reference architecture are presented resulting in an architectural model for software distribution of WfMS.

3.1 Logical View

The logical view of the architectural model [11] covers UML class diagrams and a description of class semantics, relationships and dependencies. Figure 2 shows the configuration related classes and the classes common to the whole process.

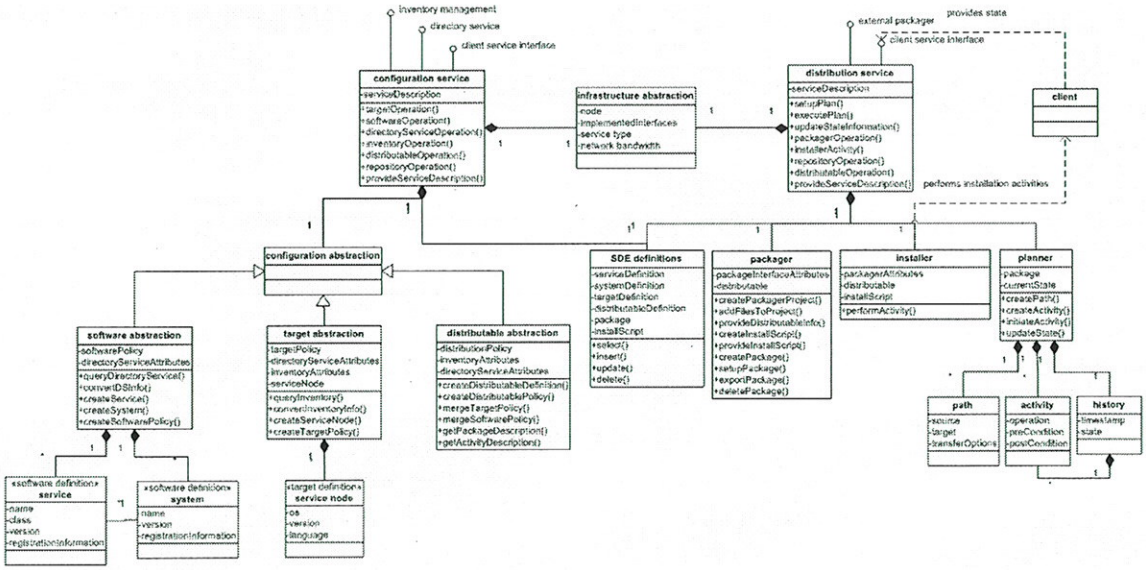

Figure 2: Model logical view – configuration

The model consists of a main class providing services to clients and external systems. It performs operations on its subclasses for the configuration process. The processes of configuration and distribution are kept separated but use common SDE definitions to share and exchange software, target and distributable data. This is reasonable because of the tight integration of distribution solutions for distributed systems like client/server products. Normally, configuration and deployment of these products is done by the same company, possibly on-site. The definitions could be separated to support physically separated processes, which is not necessary in this case. In distinction the processes of configuration and distribution are kept separated because client/server products normally have different configuration procedures but similar distribution methods. The separation should therefore provide more flexibility in the implementation. Further software and target abstractions are used to model the real-world objects. They correspond to the requirements and definitions in [1] and also consider an integration of inventory and directory service interfaces. A customer abstraction is omitted because normally one system configuration has just one customer. Most requirements like type of service can be implemented in the software abstraction and others like licensing or payment do not apply. The software abstraction is modeled with a definition of a client/server system and constituting services.

Both definitions get a name and a version attribute and provide registration information for operating system integration. These attributes are examples of many more that might be useable like a service type that specifies different service functionality within a client/server system. The software and target policies contain descriptions of supported and required services and roles of the client/server system. These terms are example descriptions that can be further extended with interface or feature definitions. The require/support semantics explicitly describe behavior that can be compared during a negotiation of the software and target policy. The most important abstraction provided by the architecture is the distributable. It covers a single item of installation and a policy about when and how to install the item. The distributable definition is a compound of software and target properties extended with properties that exclusively define the installable item itself (for example installation destination and component registration activity).

The policies and definitions of all abstractions are stored in the SDE definitions. All three abstractions are derived from a configuration abstraction class. The class provides abstract operations to describe SDE definitions with policies including dependencies, compounds and restrictions of software, target and distributable definitions' properties.

The infrastructure abstraction is used to provide information about the distribution system itself and covers physical nodes, implemented internal (clients), external interfaces (e.g. inventorying), service type, and network parameters for system maintenance.

As with configuration classes, a main service class coordinates interactions and provides interfaces to other systems. The SDE definitions are used to exchange the distributable policy and definition. The packager and the installer provide their well-known services based on a standardized interface that might have to be customized for specific products to fit. The packager creates packages from the distributable definition and the installer executes activities on the packages as defined in the distribution plan. The plan class provides the distribution process data, operations and an activity state engine. Its substructure

consists of a path class that stores physical connections that can be used during distribution and relevant data transmission options. The activity class stores specific operations that are invoked by the installer during plan execution. In general, operations that should be considered cover the whole lifecycle of distributed software comparable to the description in [1]. The installer activities can be customized with scripts or configuration files generated by an installer encapsulation class. Pre- and post conditions are used to decide whether a specific activity can be performed and has terminated accordingly. The history class is used to store state information about performed activities. This state information is also used to update the path and activity class properties to customize related activities or change execution paths as shown in the following section. To benefit from history the packager interface has to support update, adapt and repair operations. The client component shown in the diagram is external to the class infrastructure but related to the installer and distribution service for activities and state management operations.

3.2 Process View

This view provides an insight to the dynamic behavior of the client/server model. Collaboration diagrams are given preference over sequence diagrams, because structural similarities can be expressed more clearly. The focuses of the process descriptions are operations that best describe the specific requirements in client/server products; more general parts are described in [1]. First some configuration related class collaborations are shown.

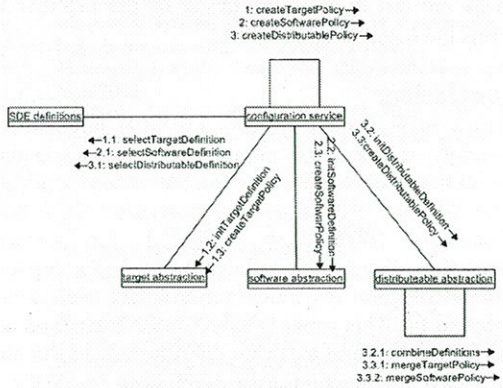

Figure 3: Model process view - configuration

Figure 3 shows the interaction between configuration classes to create a distributable definition, using target and software abstractions.

The process starts with the target abstraction, which is not a required sequence. The target definition is selected from the SDE definitions (1.1) target abstraction is initialized (1.2). Then a policy containing dependencies, compounds and restrictions of target properties is created (1.3). Those operations are derived from the abstract definitions in the configuration abstraction class. Similar operations are executed for the software definition. Selection (2.1) and initialization (2.2) in the software abstraction are performed. Then the software policy is created from properties used for software definitions (2.3).

In the third step the distributable policy is created. Therefore the distributable definition is selected (3.1) and the distributable abstraction is initialized (3.2). During this initialization the properties from the target and software definitions are combined with the distributable properties to a related definition (3.2.1). Next the distributable policy is initialized (3.3) and a merge operation for the policies is performed (3.3.1, 3.3.2). Because policies contain structured properties and rules the process is more complex and has to combine rules and properties. Additionally properties and rules of the distributable are added and new rules containing mixed properties can extend the distributable policy. A more detailed description about policy structures and content is provided in [1].

The configuration service centrally invokes these operations but the process is guided by an administrator through the client service interface. After this process the configuration task is finished and parts of the distribution process are described below.

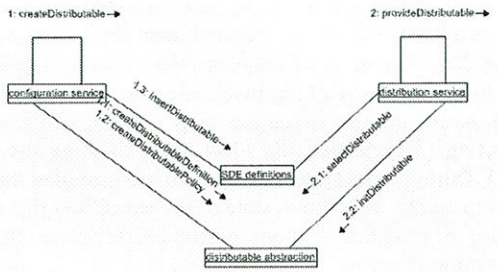

Figure 4: Model process view - shared data

Figure 4 shows the interaction of the configuration and distribution service during exchange of the distributable. The definition and policy handling (1.1, 1.2) has been described above. Instead of providing the distributable directly to the distribution process, the configuration service stores the definition in the SDE definitions (1.3). This allows the two processes to perform their operations independent of each other. The distribution service is able to define plans for distributables as soon as they become available. There is no need for a complex application level communication protocol. The distribution service selects the distributable from the SDE definitions (2.1) and the distributable abstraction is initialized (2.2). After the abstraction is set up the process can continue to define the distribution plan.

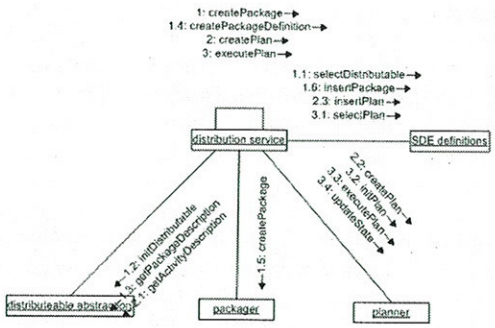

Figure 5: Model process view - distribution 1

Figure 5 shows the creation of packages and plans out of a distributable. First the distributable data stored in the SDE definitions is selected (1.1) and the distributable is initialized (1.2). In a further step, a package description is extracted (1.3). The distribution service is responsible for converting the description to a format readable for the packager (1.4) and creates a package by invoking methods of the packager interface (1.5). Finally the package is stored in the SDE definitions (1.6). In the second step of plan creation (2) the activities that can be performed on the distributable are extracted from the description (2.1) and provided during the creation step for a distribution plan (2.2). The setup of the plan also consists of ordering and selecting the required installation activities. After the plan is set up the execution step (3) can proceed by selecting (3.1), initializing (3.2) and executing the plan (3.3). During execution state updates are provided that are used to update the current state of the execution plan (3.4). Figure 6 provides a more detailed description of the distribution process.

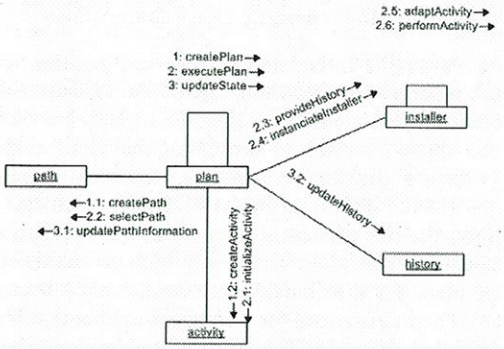

Figure 6: Model process view - distribution 2

The first phase of plan creation (1) is detailed in a path creation for destination information (1.1) and followed by an installation activity definition (1.2). The second phase of plan execution (2) consists of the initialization of a defined activity (2.1), a path selection for distributing the installer and the packages to be installed (2.2). Then the state history of the distribution plan is provided to the installer (2.3) and the installer in invoked (2.4). The installer adapts the activities based on the state information (2.5) and performs the activity on the target system (2.6). As a result of the second phase the state information of the performed activities is provided to the distribution service in the third phase (3) and influences the path information (3.1) and updates the plans activity history (3.2). Most operations provided in the process view are customized to client/server environments.

3.3 Deployment

The deployment view of an architectural model covers the physical system nodes and provides a description of how they are distributed and how they interact.

A deployment diagram of this architecture is provided in Figure 7. The model is a simplification of the deployment view provided for the Tivoli deployment system [1], which fits nicely for client/server systems such as workflow management systems presented in this paper. A development server is the starting point for the software description exchanged during initialization of the configuration process. The configuration server hosts the configuration service and the common SDE definitions. The configuration is executed in this node. The SDE definitions have to use a replication mechanism to provide the shared data to site servers where distribution takes place. The site server node normally corresponds to a customer's physical location. In large installations additional installation server are introduced to provide load balancing and fault tolerance of the process. A site server uses installation servers to provide distributables based on the path information in the distribution plan. The client as the last node is the target of the installer that performs the installation activities.

4. Conclusions

Workflow management systems (WfMS) have become increasingly relevant for intra- and interorganizational information systems as well as for Enterprise Application Integration on the Internet. Software Distribution Environments (SDEs) are required for managing configuration, distribution and deployment of components of such WfMS, but not much research has been done in that area so far. This paper is based on an evaluation of 12 products in the area of software distribution environments and distilled architectural issues relevant for SDEs for Workflow Management Systems (WfMS). We described logical and process views as well as deployment in terms of a reference architecture for a WfMS called MQ Series Workflow. As a result, this paper showed architectural considerations in terms of components, connectors and configurations of an SDE for workflow systems.

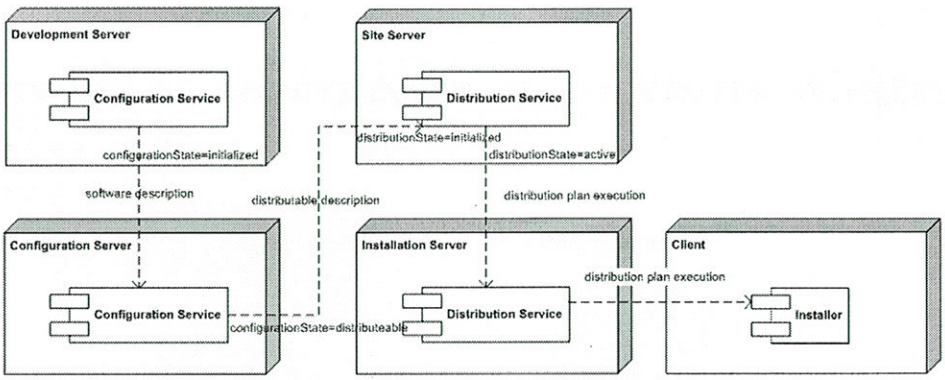

Figure 7: Client/Server model deployment view

References

[1] Anzböck, R. (2002). Architectures for a Software Distribution Environment, Master's thesis. Distributed Systems Group, Technical University of Vienna, Austria 2002

[2] Dart, S. (2000). Configuration Management: The Missing Link in Web Engineering, Artech House, London.

[3] InstallShield, InstallShield for Windows Installer Whitepaper, http://www.installshield.com/isd/resources/ 2001

[4] ManageSoft, NETDeploy Technical Specification, http://www.managesoft.com/products/technical/index.xml, 2001

[5] ManageSoft, Managing software for mobile and remote users, http://www.managesoft.com/products/mobile.xml, 2001

[6] Marimba, Marimba Management Solution, http://www.marimba.com/products/intro.htm, 1999

[7] Microsoft Corporation, DCOM, http://www.microsoft.com/com, 2001

[8] Microsoft, System Management Server Reviewer's Guide, http://www.microsoft.com/smsmgmt/ 1998

[9] Microsoft, DotNET Framework http://www.microsoftcom-/net, 2002

[10] Rational, An overview of Rational Suite ContentStudio, http://www.rational.com/products/cstudio/whitepapers.jsp, 2001

[11] Shaw, M., and D. Garlan (1996). Software Architecture: Perspectives on an emerging discipline. Prentice Hall, 1996

[12] Sun Microsystems, Enterprise Java Beans, http://java.sun.com/products/ejb/, 2001

[13] Sun Microsystems, JAVA, http://java.sun.com, 2001

[14] Sun Microsystems, Webstart, http://java.sun.com/products/javawebstart/, 2001

[15] Tivoli, Tivoli Software Distribution Factsheet, http://www.tivoli.com/news/press/pressreleases/en/2000/supplement/software_dist_factsheet.html, 2001

[16] IBM, WebSphere, http://www-4.ibm.com/software/-webservers/appserv

[17] IBM, MQ Series Workflow Workflow Concepts and Architecture, http://www-4.ibm.com/software/ts-/mqseries/workflow

[18] Workflow Management Coalition (WfMC) (1995), Workflow Management Specification Glossary, http://www.wfmc.org