Mission-Critical Business Process Management with WADE

Federico Bergenti[1], Giovanni Caire[2], and Danilo Gotta[2]

[1] Dipartimento di Matematica e Informatica, Università degli Studi di Parma,
Parco Area delle Scienze 53/A, 43124 Parma, Italy
federico.bergenti@unipr.it
[2] Telecom Italia S.p.A.
Via Reiss Romoli 274, 10148 Torino, Italy
{giovanni.caire, danilo.gotta}@telecomitalia.it

Abstract. This paper presents an overview of *WADE* (*Workflows and Agents Development Environment*), an open-source platform for *agent-based BPM* (*Business Process Management*) that has been adopted in Telecom Italia for mission-critical systems since the early 2000s. First, we sketch the main features of WADE and outline its internal architecture. Then, we describe *WANTS* (*Workflows and AgeNTS*), an industrial strength platform for large-scale network & service management that is now in service in Telecom Italia, and that leverages WADE agent-based workflows as core technology. Finally, we conclude the paper with a brief summary of the presented innovative technologies.

Keywords: Agent-based BPM, WANTS, WADE

1 Introduction

BPM (*Business Process Management*) is now a consolidated trend in IT that has recently come up as a new discipline intended to unify related topics such as Process Modeling, Workflows, Enterprise Application Integration and Business-to-Business integration.

Current BPM systems are high quality, mature tools intended primarily to manage business processes that are well structured and whose paths are identified *a priori* (see, e.g., [22]). However, the very high complexity and the intrinsic volatile and evanescent nature of today's business environment often make current BPM systems not sufficient. This has led to the identification of a number of weaknesses of current BPM systems and the criticism against available BPM systems is now a solid movement (see, e.g., [11]). We witness the rapid evolution of alternative approaches to traditional BPM that notably include *agent-based BPM systems*, and more generally, the use of the entire spectrum of *agent technologies* in the scope of BPM (see, e.g., [17]).

Besides their central role in Artificial Intelligence, as witnessed by, e.g., [20], since the establishment of *FIPA* (*Foundation for Intelligent Physical Agents*, now *IEEE FIPA Standardization Committee*) [13] in 1996 a large research community

has identified the possibilities that agent technologies provide in the design and realization of dynamic and decentralized distributed systems.

Taking the peculiar point of view on agents that FIPA promoted, we can say that agents are *loosely coupled software entities spread across a network of computation sites that communicate by means of asynchronous message passing.* Normally, the ensemble of agents that form a so called *MAS* (*Multi-Agent System*) is highly dynamic and decentralized, and the coordination of the tasks of agents is not embedded into a specific part of the system, rather it is dynamically distributed across the MAS and single agents have full control over their behaviour.

Today, agents are a mature reality in software development with notable successes, open-source reference tools, and a large literature that targets researchers and practitioners. They were used to foster collaboration (see e.g., [10]), and recently they have been used to model social networks (see, e.g., [7]), and therefore the promise of agent technologies with respect to BPM is to provide solid warranties for greater dynamism, agility, and adaptability.

Unfortunately, many barriers prevent a massive exploitation of agent technology both in terms of supporting tools and methodologies, and of the acceptance of software systems showing a certain degree of autonomy. Nevertheless several examples of deployed agent-based systems at an industrial scale exist. A number of them are described in the AgentLink site [1] and in related papers [2, 4]. In particular the trend that is combining agents, workflows, grids and SOAs [11, 14, 15, 19] appears to be very promising.

This paper presents *WADE* (*Workflows and Agents Development Environment*) [3, 23] as an innovative tool that delivers agent-based BPM in a very peculiar way. In Section 2, we present a coarse grained overview of WADE and highlight some of its most interesting features showing how WADE approaches agent-based BPM. Here we mainly focus on the features that make WADE a good candidate to develop mission-critical systems with complex internal logics, and we do not get into the details of other interesting aspects of this technology. Then, in Section 3, we present a real-world utilization of WADE in large-scale network management. In particular, we present *WANTS* (*Workflow and AgeNTS*) and we emphasize the use of WADE workflows in it.

2 The WADE Platform

In broad terms, a *workflow* is the static definition of a process in terms of activities to be executed, relations between them, criteria that specify the activation and termination of activities, additional information, such as the participants, the software tools to be invoked, required inputs and expected outputs, and internal data manipulated during the execution.

Nowadays the workflow metaphor is commonly used in BPM and a workflow represents in this area a possible, and probably the preferred, means for describing a business process.

The main advantage of implementing a process as a workflow is the inherent expressiveness of the *workflow metaphor*. In fact, a workflow can be represented in a purely graphical form that is understandable by domain experts as well as by programmers. Domain experts can validate system logics directly and not only on documents, that are often not timely updated.

Another important characteristic of workflows is that the steps that compose the process are clearly and explicitly identified. This enables creating automatic mechanisms that trace the execution of a workflow, thus facilitating system monitoring and problem investigation.

Additionally, when processes have to be executed within the scope of a transaction, the explicit identification of workflow steps allows using semi-automatic rollback procedures that can be activated in case of unexpected faults.

Finally, since workflows are fully self-documented, workflow-based development releases the development team of the burden of keeping documentation aligned with running software.

2.1 WADE as a BPM platform

WADE (*Workflows and Agents Development Environment*) [3, 23] is an open source platform designed to allow the encapsulation of workflows into agents, and it is designed to promote the synergy between agent and workflow metaphors.

WADE is essentially the main evolution of *JADE* (*Java Agent and DEvelopment framework*) [5, 6, 9, 8, 16], which is a popular open source framework that facilitates the development of interoperable multi-agent systems.

JADE has been used in many research and industrial systems at an international scale since its initial development back in 1998. Just to cite a known and appreciated use of JADE, which is also related to the network management scenario discussed in Section 3, BT uses JADE as the core platform for *mPower* [18], a multi-agent system that is used by BT engineers to support cooperation between mobile workers and team-based job management.

As depicted in Figure 1, WADE mainly adds to JADE the support for the execution of tasks defined according to the workflow metaphor, and it also provides a number of mechanisms that help managing the inherent complexity of a distributed system both in terms of administration and fault tolerance.

In this work we focus on the aspects related to workflow-based development because we consider them the most characterizing feature of WADE in its current form. Nonetheless, it is worth noting that WADE can be used as an everyday development platform and, in principle, developers can use WADE with little or no adoption of workflows. As we often say, WADE supports *notepad programming* because the full power of the platform is accessible from plain old Java objects. However, remembering that one of the main advantages of the workflow metaphor is the possibility of representing processes in a friendly graphical notation, WADE comes with a development environment called *WOLF* (*WOrkflow LiFe cycle management environment*) [12] that facilitates the creation of WADE workflow-based agents. WOLF provides users with a friendly graphical notation

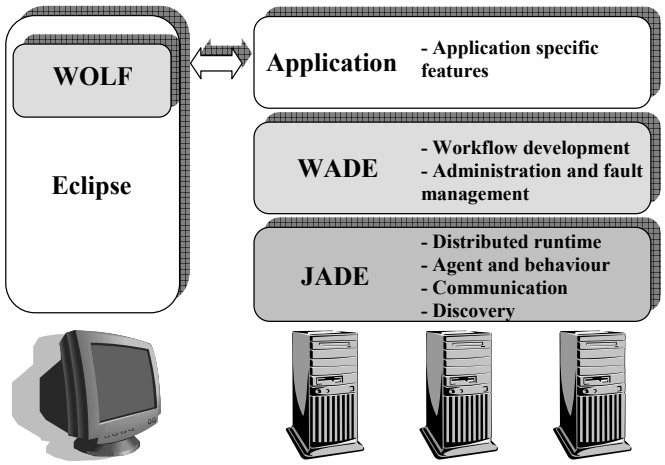

Fig. 1. A WADE-based system and its major components.

and an advanced graphical editor integrated in the Eclipse IDE, and little or no programming skills are needed to implement simple workflows.

In fact, the main challenge that drives and motivates the design of WADE is to bridge the gap between the BPM-level use of workflows and their use to implement the internal logics of a distributed system. That is, even if it could be used for that purpose too, WADE does not only target high-level orchestrations, rather it provides concrete support for the implementation of the internal behaviour of the system. Furthermore, unlike the majority of existing workflow-based systems that provide a powerful centralized engine, in WADE each agent embeds a *micro-workflow engine* and a complex process can be carried out by a set of cooperating agents, each one executing a piece of the process.

Another key characteristic of WADE is that it does not adapt a privileged textual or graphical notation to express workflows, rather WADE sees a workflow as a set of Java classes. This makes the workflow immediately executable and no interpretation or just-in-time compilation is needed. Moreover, this choice eases the graceful scaling from the BPM-level down to the level of the internal logics. WOLF is an essential tool in this picture because it provides a convenient graphical view of workflow classes and it smoothly integrates workflow editing and Java editing with the advanced features that the Eclipse IDE provides.

2.2 Workflow meta-model

In order to facilitate import/export operations from/to workflow standard representation formalisms, WADE adopts a workflow meta-model closely derived from that defined by the Workflow Management Coalition for XPDL [24, 25]. The main elements that compose such a meta-model are described hereafter.

A task that is being described is called a *process*. A process is composed of a set of *activities* each one corresponding to the execution of given operations. A process defines a single *start activity* (specifying the execution entry point) and one or more *end activity* (specifying the execution termination points). Each non-end activity has one or more outgoing *transitions*, possibly associated with a condition, leading to another activity in the process. Once the execution of the operations included in a given activity is terminated, the conditions of all outgoing transitions are evaluated. As soon as a condition holds, the corresponding transition is fired and the execution flow continues with the operations included in the destination activity.

A process can have one or more *formal parameters* defining the type of required inputs and expected outputs. At process invocation time proper values must be provided for input parameters and, at the end of the execution, the values produced as output parameters are returned to the requester.

WADE supports several types of activity and number of the different types of activity is constantly growing as the use of the platform increases. Besides many useful types of activity, the most important types are as follows.

- *Subflow activities.* The operations included in a subflow activity consist in the invocation of another workflow process. The execution of the subflow takes place in a separate computational space and can be even carried out by a different agent (possibly running in a remote host).
- *Tool activities.* The operations included in a tool activity consist in invoking one or more external tool generically identified as *applications*. Applications are computational entities defined outside the workflow process and wrapped by a uniform interface.
- *Code activities.* The operations included in a code activity are specified directly by a piece of Java code embedded in the workflow process definition. It should be noticed that, unlike tool and subflow activities, code activities do not belong to the XPDL meta-model and are a proprietary WADE extension.

3 The WANTS Platform

With over 8.95 million broadband connections (retail and wholesale) in 2012, Telecom Italia is currently one of the leading operator in the Italian TLC market. It has one of the most penetrating and advanced networks in Europe, with an extension of over 114 million km in copper lines and 5.7 million km of optical fibers [21].

Considering the huge business volumes involved in this scenario, it is not difficult to understand that network management systems carrying out everyday intensive operations have strong requirements in terms of scalability, robustness and flexibility. *WANTS* (*Workflow and AgeNTS*) is a platform for the management of telecommunication networks and services that Telecom Italia is currently

using to manage its broadband network for many *OSS* (*Operation Support System*) and *BSS* (*Business Support System*) activities. In brief, some of the key architectural point of WANTS are:

- WANTS is a *network & service management* platform based on distributed agents running hierarchical workflows;
- WANTS has a documentation inventory of all processes and all equipment information models;
- WANTS has a model inventory holding all process descriptions and network resource information models with automatic synchronization with all operating functionalities of the platform, which solves the problem of having a continuous up-to-date documentation of equipment modeling and operating processes;
- The architecture of WANTS is intrinsically adaptive in detecting and predicting saturation using observation of real load and managing resource utilization.

The prior art in network & service management platform implements flexibility as more or less sophisticated configuration capabilities and development environments that helps system designer in building the skeleton of new modules to support new services or new technologies. This degree of flexibility is certainly not enough and new trends recommend extraction of embedded process logic from components in order to have an external process manager that can orchestrate the flow of actions, thus achieving a much greater flexibility.

WANTS pushes such an approach even further because all process logic that remains inside each component is programmable from an external management entity, using workflow engines in all components of the platform. This concept is radically conceived so that each component is essentially a workflow engine that runs each kind of process, moving further from current approaches of platforms where each component runs a specific domain functionality to a platform where each component can be freely focused on particular domain functionalities needed by current policies, availability of resources, and load status.

Figure 2 outlines the major components of the WANTS architecture. Each *PA* (*Protocol Adapter*) is responsible for interfacing all the network equipments of a designated area that offers the same API or protocol, e.g., SNMP, telnet, or TL1. Each PA offers, as services to *RPs* (*Resource Proxies*), the execution of basic operation on the equipment.

Each RP is responsible for creating, maintaining and managing a so called *image* of a single equipment. The image is a representation of the configuration of the equipment according to a defined information model. The alignment of the image to the actual network is done by means of periodic checks or by means of proactive notification from PAs or from equipments themselves.

Each RP performs activities typical for the RP level: such activities are called *layer 3* activities and they can be structured in sublayers. Activities at the top of layer 3 can be externally invoked, thus they are the services that RP offers to the *AAs* (*Agent Applications*) and to external applications. They represent the operations that can be atomically performed on the equipment that the RP

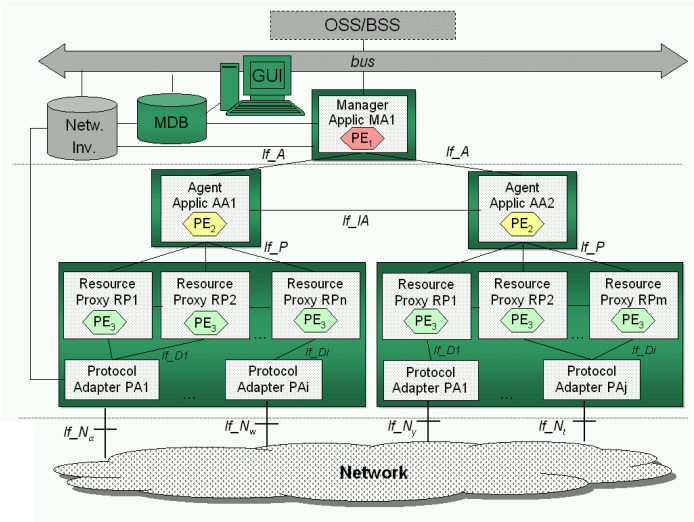

Fig. 2. Major elements of the WANTS architecture.

manages. Examples of services offered by RP are: configure port, create cross-connection, modify connection attribute. Each of such processes can include a sequence of basic commands to be sent and/or received to/by the equipments. Activities at the bottom of layer 3 uses services offered by PAs.

The image handled by an RP is dynamically defined by the information model of the represented resource. This model is distributed by the *MA* (*Manager Application*) loaded by the RP and then instantiated with values retrieved by the resource. In this manner, changes and additions of information models do not require software changes in the component of the platform, thus allowing a high degree of flexibility, as long as the equipment API or protocol are supported by the PA.

The *network inventory*, as a fundamental component of any network management platform, is split into two concepts in WANTS: a *DNI* (*Distributed Network Inventory*) and a *CNI* (*Centralized Network Inventory*). The former is the collection of all images contained in all the RPs; its usage is for all real-time (or almost-real-time) tasks such as provisioning, assurance, and performance, where having updated information on the configuration and state of the network is necessary for the accuracy and effectiveness of the task.

The latter type of network inventory is the CNI: it is basically the usual network inventory component. In WANTS this inventory is used only for non-real-time tasks where continuous updates with the network are not possible because of the centralized design. Nonetheless, the CNI is periodically updated interacting with the RPs.

The *MDB* (*Model Data Base*) is a documentation inventory of all processes and all equipment information models. The MDB is kept synchronized with the processes and information models running in the architecture. This represents another major benefit for the network operator that does no longer need to dig information from huge amount of documentation of the different component vendors with the risk of not finding or even finding obsolete documentation.

AAs are agents that are in charge of performing workflows for coordination of a set of RPs and for the execution of activities typical of the agent level: these activites are called of *layer 2* and they can be structured in sub-layers. Activities at the top of layer 2 can be externally invoked, thus they are the services that AAs offers to the MA. Each AA can perform any type of process of layer 2 or, in other words, supports all *FCAPS* (*Fault, Configuration, Accounting, Performance, Security*) functionalities.

AA interact among each other via a *community protocol* to support distributed execution of management functionalities like, e.g., distributed circuit design.

AAs do not require software update to support new services and technologies because of the extreme flexibility of the processes that are received by the MA that are loaded and executed by the AA. Each AA is responsible for a local performance monitoring to inform the MA about performance status.

The MA is responsible for the following tasks:

- Manage the distribution of processes of layer 2 and 3 to AAs and RPs retrieving the process definitions from the MDB;
- Manage the distribution of equipment information models for the RPs retrieving the equipment information models from the MDB;
- Monitor the state of the platform with information provided by the AAs, included distribution of components, domain management (partitioning of whole network among the AAs), performance monitoring and consequent actions like load balancing between AAs.
- Interact with external systems, like other legacy OSS or BSS.
- Execute activities typical of *layer 1*, that are meant to provide functionalities that require interaction with external entities (other than AAs) or coordination among agents that cannot easily or efficiently be performed by AAs through the community protocol.

It is worth noting that besides AAs, all executors of any of the three layers are developed as a combination of a workflow to fully exploit the power of the underlying WADE platform.

The mobility service that WADE transparently provides is useful to move agent across hosts in order to solve agent deployment and fault tolerance issues. If an agent unexpectedly terminates, WADE instantiates a new agent and moves it toward the target host (obviously if the host is still running) in order to replace defunct agent. The MA periodically monitors the presence of AA agents and acts accordingly. Moreover, agent mobility is also useful to move agent across hosts to face load balancing issues. This can occur if an AA, for instance, is continuously requested by an MA to run a workflow. The AA quickly becomes a bottleneck

and WANTS either instantiates a new AA and moves it toward the host where overloaded AA run, or it simply moves the overloaded agent on a host with lower CPU load. The AA agent communicates the current overload condition to the MA, and after that the MA requests for agent mobility.

4 Conclusions

This paper presents a set of tools that are in routine work in Telecom Italia to manage mission-critical business processes. The first tool is WADE, an open-source platform that can be freely used to implement sophisticated systems that rely on the synergy between agent and workflow metaphors. The second tool is WANTS, a mission-critical network & service management platform completely developed in Telecom Italia using WADE and currently deployed in the field.

WANTS has a direct influence on the work of thousands of technicians and on millions of customers and, as a consequence, it has strong requirements in terms of scalability and flexibility. The enabler for this compelling price/performance proposition is WADE that uses the combination of agents and workflows to achieve:

- High flexibility in defining and modifying services;
- Deep control on the accuracy of results in a fault tolerant environment;
- High performance and scalability;
- High robustness and user-friendliness;
- High control and maintainability on the logics used in the platform.

In order to bring the workflow approach from the business process level down to the level of system internal logics, WADE allows viewing workflows both in terms of high-level process descriptions and of lower-level Java code. This allows combining the flexibility of the Java language and the power of the Eclipse IDE with the expressiveness and traceability of workflows. Moreover, it lowers the barrier of acceptance of the presented technology from both domain experts, that are familiar with the graphical notation, and from technical personnel, that appreciates working with a well-known language like Java.

Unfortunately, the fuzzy buzzwords that surround agent technology, like *autonomy* and *self-consciousness*, still encounters some resistance especially in a large enterprise like Telecom Italia.

References

1. AgentLink III Web site. `http://www.agentlink.org`
2. AgentLink III. *Agent Technology Roadmap*. Available at `http://www.agentlink.org/roadmap/index.html`
3. Banzi M., Caire G., Gotta D. WADE: A software platform to develop mission critical, applications exploiting agents and workflows. Procs. *Int'l Conf. Autonomous Agents and Multi-Agent Systems*, 2008.

4. Belecheanu R., Munroe S., Luck M., Payne T., Miller T., Pechoycek M., McBurney P. Commercial applications of agents: lessons, experiences and challenges. Procs. *Int'l Conf. Autonomous Agents and Multi-Agents Systems*, 2006.
5. Bellifemine F., Caire G., Greenwood, D. *Developing Multi-Agent Systems with JADE*. Wiley Series in Agent Technology, 2007. ISBN 978-0-470-05747-6.
6. Bellifemine F., Poggi A., Rimassa G. Developing multi-agent systems with a FIPA-compliant agent framework. *Software: Practice & Experience*, 31:103–128, 2001.
7. Bergenti F., Franchi E., Poggi A. Selected models for agent-based simulation of social networks. Procs. *Symp. Social Networks and Multiagent Systems*, 2011.
8. Bergenti F., Poggi A. Ubiquitous information agents. *Int'l J. Cooperative Information Systems*, 11(3–4):231–244, 2002.
9. Bergenti F., Poggi A., Burg B., Caire G. Deploying FIPA-compliant systems on handheld devices. *IEEE Internet Computing*, 5(4):20–25, 2001.
10. Bergenti F., Poggi A., Somacher M. A collaborative platform for fixed and mobile networks. *Communications of the ACM*, 45(11):39–44, 2002.
11. Buhler P.A., Vidal, J.M. Towards adaptive workflow enactment using multiagent systems. *Information Technology and Management*, 6(1):61-87, 2005.
12. Caire G., Quarantotto E., Porta M., Sacchi G. WOLF - An Eclipse plug-in for WADE Procs. *IEEE Int'l Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*, 2008.
13. FIPA Web site. `http://www.fipa.org`
14. Foster I., Jennings N. R., Kesselman C. Brain meets brawn: Why grid and agents need each other. Procs. *Int'l Conf. Autonomous Agents and Multi Agent Systems*, 2004.
15. Greenwood, D., Callisti, M. Engineering Web service-agent integration. Procs. IEEE Int'l Conf. Conference of Systems, Man and Cybernetics, 2004.
16. JADE (Java Agent DEvelopment framework) Web site. `http://jade.tilab.com`
17. Jennings N. R., Faratin P., Johnson M. J., Norman T. J., Wiegand, M. E. Agent-based business process management. *Int'l J. Cooperative Information Systems*, 5:105–130, 1996.
18. Lee H., Mihailescu P., Shepherdson J. Realising team-working in the field: An agent-based approach. *IEEE Pervasive Computing*, 1:85–92, 2007.
19. Negri A., Poggi A., Tomaiuolo M., Turci P. Dynamic grid tasks composition and distribution through agents. *Concurrency and Computation: Practice and Experience*, 18(8):875–885, 2006.
20. Newell, A. The Knowledge Level. *Artificial Intelligence*, 18(1):87–127, 1982.
21. Telecom Italia S.p.A. *Relazione Finanziaria Annuale 2012*. Available at `http://www.telecomitalia.com`
22. Trione L., Long D., Gotta D., Sacchi G. Wizard, WeMash, WADE: Unleash the Power of power of collective intelligence. Procs. *Int'l Conf. Autonomous Agents and Multiagent Systems*, 2009.
23. WADE (Workflows and Agents Development Environment) Web site. `http://jade.tilab.com/wade`
24. WFMC (WorkFlow Managment Coalition) Web site. `http://www.wfmc.org`
25. XPDL (XML Process Definition Language) Web site. `http://www.wfmc.org/standards/xpdl.htm`