WADE: A software platform to develop mission critical applications exploiting agents and workflows

Giovanni Caire
Telecom Italia
Via Reiss Romoli 274
10148 Torino - Italy
+39 011 2286107
giovanni.caire@telecomitalia.it

Danilo Gotta
Telecom Italia
Via Reiss Romoli 274
10148 Torino - Italy
+39 011 2288061
danilo.gotta@telecomitalia.it

Massimo Banzi
Telecom Italia
Via V. Zambra 1
38100 Trento - Italy
+39 0461 316408
massimo.banzi@telecomitalia.it

ABSTRACT

In this paper, we describe two mission critical applications currently deployed by Telecom Italia in the Operations Support System domains. The first one called "Network Neutral Element Manager" implements a mediation layer between network elements and OSS systems. The second one, known as "Wizard", provides step-by-step guidance to technicians performing maintenance operations in the fields.

Both applications have strong requirements in terms of scalability and flexibility and exploit the combination of agents and workflows to meet them. As such both of them are based on a common software platform called WADE (Workflows and Agents Development Environment). WADE is the main evolution of JADE a popular Open Source framework that facilitates the development of interoperable intelligent multi-agent systems. WADE adds to JADE the support for the execution of tasks defined according to the workflow metaphor and a number of mechanisms that help managing the complexity of the distribution both in terms of administration and fault tolerance. In this paper in particular we focus on the workflow aspect and we show how WADE tries to bring the workflow approach from the business process level to the level of system internal logic.

Categories and Subject Descriptors

I.2.11 {**Artificial Intelligence**]: Distributed Artificial Intelligence - *Multiagent systems*; C.2.4 {**Computer Communication Systems**]: Distributed systems; D.2.11 [**Software Engineering**]: Software architecture

General Terms

Management, Performance, Languages.

Keywords

Software Agent, workflow, JADE, Open Source, XPDL, OSS, Telecommunication network, Scalability, Flexibility.

Cite as: WADE: A software platform to develop mission critical applications exploiting agents and workflows, Caire G., Gotta D., Banzi M., *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008) – Industry and Applications Track*, Berger, Burg, Nishiyama (eds.), May, 12-16., 2008, Estoril, Portugal, pp. 29-36.

1. INTRODUCTION

With 7.3 million broadband connections (retail and wholesale) [1], Telecom Italia is currently the leading operator in the national TLC market. It has one of the most penetrating and advanced network in Europe, with an extension of over 107 million Km in copper lines (access) and 3.8 million km of optical fibers (transport and access).

Recently the deployment of the passive optical fiber network (GPON) in combination with VDSL2 modulation supporting transmission rates up to 50 Mbit/s enabled the provisioning of more advanced services such as high definition television, telemedicine, and so on.

Considering the huge business volumes involved in the described scenario, it is not difficult to understand that management systems carrying out everyday intensive operations have strong requirements in terms of scalability, robustness and flexibility. In this paper in particular we describe two mission critical applications currently deployed by Telecom Italia in the Operation Support System (OSS) domains that exploit the agent paradigm and the workflow metaphor to meet such requirements.

Both applications are built upon a common software platform called WADE (Workflows and Agents Development Environment). WADE is the main evolution of JADE ([2], [3], [4], [5]) a popular Open Source framework that facilitates the development of interoperable intelligent multi-agent systems. For instance British Telecom uses JADE as the core platform for mPower [6], a multi-agent system that is used by BT engineers to support cooperation between mobile workers and team-based job management. WADE adds to JADE the support for the execution of tasks defined according to the workflow metaphor and a number of mechanisms that help managing the complexity of the distribution both in terms of administration and fault tolerance. In this paper in particular we focus on the workflow aspect and we show how WADE tries to bring the workflow approach from the business process level to the level of system internal logics

Many barriers preventing a massive exploitation of agent technology remains both in terms of supporting tools and methodologies and of acceptance of software applications showing a certain degree of autonomy and self-consciousness. Nevertheless several examples of deployed agent-based systems for industrial application exist. A number of them are described in the Agent Link site [7] and in related papers [8]. In particular the trend that is mixing agents, workflows, grid and SOA ([9], [10],

[11], [12], [13], [14]) appears to be very promising and WADE fits in it.

The paper is structured as follows: in chapters 2 and 3, we present the applications mentioned above that have a direct influence on the work of thousands of technicians and potentially millions of customers. The first one called "Network Neutral Element Manager" implements a mediation layer between network elements and OSS systems. The second one, known as "Wizard", provides step-by-step guidance to technicians performing maintenance operations in the fields. In chapter 4 and 5 we focus on WADE, the software platform at the basis of both applications actually implementing the agents and workflows related features. Finally in chapter 6 we draw some conclusions and present future activities.

2. NETWORK NEUTRAL ELEMENT MANAGER

One of the major problems in the Operation Support Systems domain is related to the lack of standards in the management interfaces that network elements provide. This is even more critical in large and highly multivendor telecommunication networks such as that of Telecom Italia. Because of this lack of standardization each vendor provides its own Element Manager with a proprietary interface both in terms of protocol (CORBA, SNMP, TL1, XML being the most common ones), mimic of interactions and data modeling. As a consequence all OSS systems that require communicating with the network (such as the activation system, the troubleshooting system, the performance monitoring system and so on) need to embed proper adapters for each vendor and type of device. Moreover every time a new technology, a new vendor or even a new release of a network element is deployed, all OSS systems need to be updated thus multiplicating the effort and slowing down the roll out of new services.

In order to face this problem, a project called "Network Neutral Element Manager" was started four years ago in the OSS Innovation department of Telecom Italia. The focus of the project was the development of a mediation layer decoupling the management systems from the network elements. As depicted in Figure 1 the Network Neutral Element Manager (NNEM) aimed at hiding the diversity of the underlying technologies, vendors and types of device to OSS systems by providing a uniform north-bound interface.

Another important benefit of the NNEM was related to the possibility of controlling the management overload. Having a single entity carrying out all interactions with the network, in facts, allows governing the requests of the different OSS systems. This avoids slowing down the performances of the network due to uncontrolled accesses performed by completely un-coordinated systems.

Considering the challenging goals described above, it was clear that the NNEM had to meet strong requirements in terms of:

Scalability - The NNEM had to be able to manage thousands of network elements and serve several management systems potentially producing each one thousands of requests per minute.

Flexibility - A big telecommunication network is a sort of "living animal" that evolves mostly every day in terms of new services, new technologies, new vendors, new types of device and new firmware releases. Clearly the NNEM had to cope with this continuous evolution providing proper mechanisms to support hot deployment of new/modified system logics.

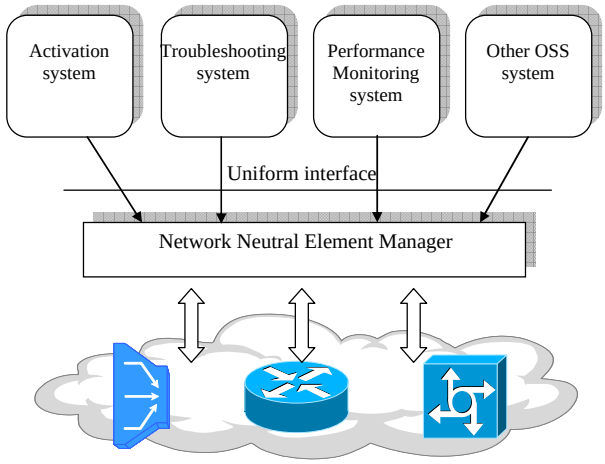

Figure 1. The Network Neutral Element Manager

In order to meet the scalability requirement it was decided to adopt agent technology as the basis for the NNEM. In particular the core of the system was architected with one agent called Resource Proxy for each network element. A Resource Proxy is responsible for virtualizing the related network element and managing all accesses to it. Each Resource Proxy keeps an image of both the physical structure (cards, ports and so on) and the configurations (logical interfaces, cross-connections, profiles) of the virtualized device. This image, called "cache" is normalized according to a vendor independent model closely derived from the SID (Shared Information/Data model) being defined within the scope of the Tele Management Forum [25]. The Resource Proxy also processes the traps issued by the virtualized device and is therefore able to keep its cache constantly up to date.

Flexibility was achieved by describing all the logics of interactions with the network elements to be carried out by Resource Proxy agents as workflows. That is, instead of directly embedding the code implementing these interactions, Resource Proxy agents were designed to include a very light workflow engine. Each time a modification occurs in the network a new/modified workflow reflecting the modification can be deployed at runtime thus making the Resource Proxy agents immediately able to cope with the new situation.

Both the basic agent-related features such as execution model, communication, discovery and life cycle management and the ability to execute possibly long and complex tasks defined according to the workflow metaphor are provided to the NNEM system by a software platform called WADE that will be described in chapter 4.

The Network Neutral Element Manager is currently deployed on 15 low cost HP Proliant DL 145 servers each one equipped with 2 ADM Opteron 246 processors (2 GHz) and 4 Mbytes of RAM (cost per unit between 2000 and 3000 euros at the end of 2006). It manages the network elements of the IP broadband and ultra-broadband access (~2000 devices considering IP DSLAM and ONU from 4 different vendors) serving the activation,

troubleshooting and partially the fault management processes. Extensions to new domains (and in particular the GBit Ethernet metropolitan network) and new processes (such as performances monitoring and configuration management) are under evaluation.

As described in [26] laboratory tests carried out in 2007 shown that more than 5400 IPTV service activation requests per hour can be served on just 3 of the servers mentioned above.

3. WIZARD

The costs of the operational processes, such as the ones for providing new services to customers or for removing failures and malfunctions, represent an important percentage of the costs that telecommunications operators must face yearly. Hence the importance of systems aimed at reducing such costs through tools supporting activities of both operator workforce and customers, which can thereby be directly involved in removing troubles related to equipments in the network and in the customer home.

To support the workforce that is directly involved in actions of repairing failures/malfunctions of the network we have realized a software system called Wizard. Wizard guides the technical staff in a complete, integrated and exhaustive way, through all the steps to be followed in problem-solving activities. A complete guide enables both a reduction in the working times for the technicians and a faster insertion of technicians that are new to the job.

In the first place, the system provides a direct interaction with the systems/platforms responsible for network and service management. This significantly reduces the times of execution of problem-solving activities since the correct completion of the jobs performed by the technical staff can be verified in real-time by Wizard that proactively triggers suitable checks with the right data on the relevant OSS systems.

In the second place, the system represents using formal tools (such as the workflows) the operative knowledge to be shared by technicians. This enables a further reduction of the working times of the technicians through an unambiguous and readily understandable description of the activities to be carried out. Such formal representation would also avoid any possible difficulties of interpretation of the supplied indications, which can lead technicians to execute activities that are useless or even harmful for the network.

The Wizard system has been developed on top of WADE (that will be described in chapter 4) and is aimed to support business processes including both automatic tasks (machine to machine interactions) and human task (human workflows, that means the support for human activities and a real-time interaction with the user). One of the features added by Wizard is the concept of Workflow Driven GUI, that means a Graphic User Interface that allows the real-time interaction between the workflow execution and the user (see Figure 2).

This GUI runs also on mobile assets in order to take into account nomadic workplace environments

We have carefully chosen the case study working together with on field technicians in order to understand all details, critical points and bottleneck of their job.

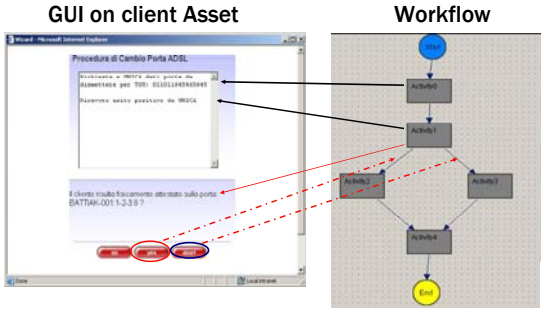

Figure 2. Workflow driven GUI

One result is a new way to perform the ADSL diagnosis that reduce the work for the back office and empower the on field technician.

The technician on field receives the work-request on his mobile asset, drives to the central office (where the network elements are located) and starts through his mobile asset the “ADSL diagnosis and repair“ workflow that drives him through all steps of the whole process interacting with him and with the remote systems when needed (see Figure 3)

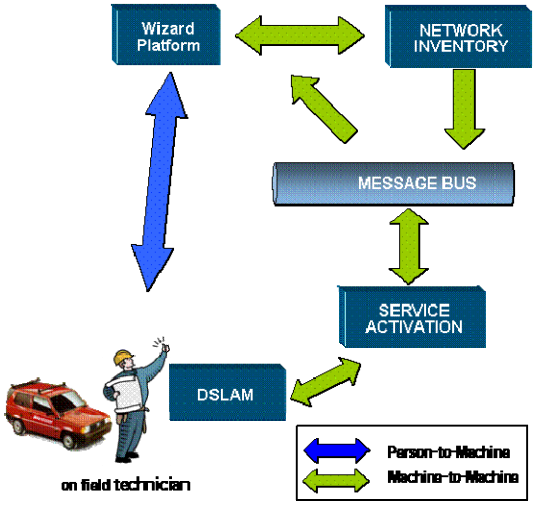

Figure 3. ADSL Diagnosis and repair scenario

At the end of the workflow the technician gets, as a result, the new pair position on MDF (main distribution frame) related to the new DSLAM port to the technician

Finally the technician moves the jumpers on the new MDF positions and closes his work-request

The diagnosis process now is well defined and documented as part of “ADSL Diagnosis and repair workflow”

Technicians of any experience level are able to address the work-request and possibly even learn through the workflow they execute in order to accomplish it

The technician through the Wizard Platform can use the automatic service fulfillment chain in order to reconfigure the circuit saving a lot of time (from hours to minutes)

The feedback from the field is really good because of the time saved by the technician avoiding the phone call to the back office operator and the time saved by the back office operator itself . The overall process now takes only few minutes instead of hours

The solution presented in this section is used in Telecom Italia by hundreds of technicians each day guaranteeing the assurance on the 7.3 million broadband connections.

4. THE WADE PLATFORM

Though addressing different domains and showing opposite characteristics in terms of user interactivity, the applications described in sections 2 and 3 are both built on top of a common software platform called WADE. WADE (Workflow and Agent Development Environment) represents the main evolution of ***JADE*** [5], a popular open source middleware conceived to facilitate the development of distributed applications based on the ***agent-oriented paradigm***.

As depicted in Figure 4, JADE provides a distributed runtime environment, the "agent" and "behaviour" abstractions, peer to peer communication between agents and basic agent lifecycle management and discovery mechanisms. WADE adds to JADE the support for the execution of tasks defined according to the ***workflow*** metaphor and a number of mechanisms that help ***managing the complexity of the distribution*** both in terms of administration and fault tolerance. This paper in particular focuses on the aspects related to workflow based development that we consider WADE most characterizing feature.

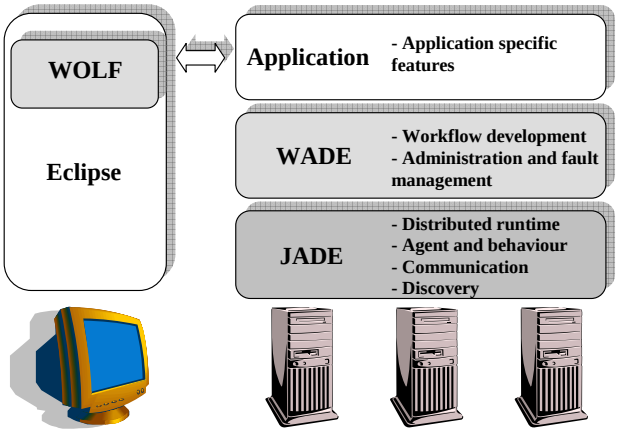

Figure 4. The WADE platform

In principle WADE supports "notepad-programming" in the sense that there is no hidden stuff that developers can't control. However, especially considering that one of the main advantages of the workflow approach is the possibility of representing processes in a friendly graphical form, WADE comes with a development environment called ***WOLF*** that facilitates the creation of WADE-based application. WOLF is an Eclipse [15] plug-in and as a consequence allows WADE developers to exploit the full power of the Eclipse IDE plus additional WADE-specific features.

4.1 Workflow based development

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

The main advantage of implementing a process as a workflow is the expressiveness of the workflow metaphor. ***A workflow in facts can be represented in a purely graphical form*** that is understandable by domain experts as well as by programmers. Domain experts can therefore validate system logics directly and not only on documents that most of the time are not perfectly up to date. In some cases they could even contribute to the actual development of the system without the need for any programming skill.

Another important characteristic is that the steps that compose the process are explicitly identified. This makes it possible to create ***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, ***semi-automatic rollback procedures*** can be activated in case of unexpected fault.

Finally, since ***workflows are fully self-documented***, workflow-based development releases the development team of the burden of keeping documentation aligned each time design choices must be revisited to face implementation details.

4.1.1 Scope

Nowadays the workflow metaphor is mostly used in BPM (Business Process Management) environments where a workflow represents a business process and orchestrates a number of existing systems typically (but not necessarily) accessible by means of Web Services-based interfaces.

The main challenge in WADE is to ***bring the workflow approach from the business process level to the level of system internal logics***. That is, even if it could be used for that purpose too, WADE does not target high level orchestration of services provided by different systems, but the implementation of the internal behaviour of each single system.

A direct consequence of the described approach is that WADE is expected to be particularly suitable for applications that imply the execution of ***possibly long and fairly complex tasks***.

Furthermore, unlike the majority of existing workflow systems that provide a powerful centralized engine, in WADE each agent can embed 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.

From an industrial point of view one advantage in using WADE is the possibility to develop mission critical applications that can work on grid of blade servers (or PC) with great scalability, and big savings in hardware [16]. For example, one impressive success of PC-derived components, harnessed in parallel by open-source-based software, is the Google search engine, implemented on a massive cluster comprising more than 15,000 commodity-class PCs as described in an paper published in 2003 [17]. The

Google application has been designed to take advantage of these affordable building blocks, with different queries running on different processors and with a partitioned index that allows single queries to run on multiple processors. They make it possible for Google to pursue the lowest possible ratio of price to performance and not, in the manner of past supercomputer efforts, peak processor performance regardless of cost.

4.1.2 Approach

As mentioned the workflow metaphor provides a clear and intuitive representation of the process execution flow. On the other hand a purely graphical or descriptive formalism (such as BPMN, BPEL, WS-BPEL ([22], [23], [24]) is not suitable to specify all the details involved in a process that implements a piece of the business logic of a given software system. A usual programming language such as Java is definitely more powerful and flexible to deal with data transformations, computations and other low level auxiliary operations that can be needed during the process execution. Furthermore programmers used to exploit powerful Integrated Development Environments such as Eclipse would not even consider working with a platform that does not provide the same level of support in terms of searches, navigations, error reporting, automatic suggestions, refactoring, debugging and so on.

Taking into account the above considerations, the approach followed by WADE is to provide a workflow view on top of a normal Java class. That is a workflow that can be executed by WADE agents is expressed as a Java class with a well defined structure (detailed in section 5.2.1). As such WADE workflows can be edited, refactored, debugged and in general managed as all Java classes. In addition of course the execution flow they specify can be presented and modified in a friendly, graphical way. More in details Wolf (the development environment for WADE based applications) is an Eclipse plugin and allows developers to work with a graphical view (suitable to manage the process flow) and a code view (the usual Eclipse Java editor suitable to define execution details) that are kept in synch.

Therefore the WADE micro workflow engine does not embed an interpreter of a workflow description language, but just executes compiled Java code. This on the one hand makes it extremely performant, but on the other hand requires the necessary workflow classes to be available when an agent is requested to execute a workflow. For this reason WADE uses ad hoc Java class loaders to allow deploying new/modified workflows that become immediately executable without the need to turn the system down.

5. Workflow representation formalism

As mentioned in section 4.1.2, WADE provides a workflow view on top of normal java classes that can therefore be managed exploiting the full power of the Eclipse IDE. In this section we give more details about how a java class representing a workflow that can be executed by WADE agents looks like.

5.1 The meta-model

In order to facilitate import/export operations from/to workflow standard representation formalisms, WADE adopts (when relevant) a workflow meta-model closely derived from that defined by the by the Workflow Management Consortium for the **XPDL** language ([18], [19], [20], [21]). The main elements that compose this 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 to 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 goes on 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.

Depending on the included operations, there are different types of activity the most important being.

- ***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.
- ***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 (as will be described in section 5.3) can be even carried out by a different agent (possibly running in a remote host) with respect to that performing the main process.
- ***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.

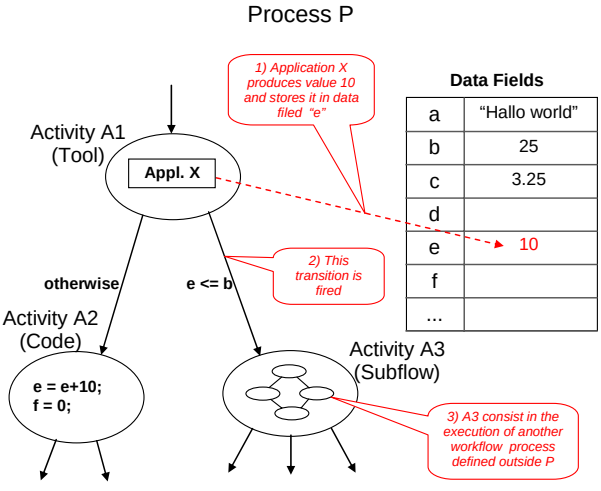

Figure 5. Main elements in the WADE meta-model

Finally the process makes reference to a set of ***Data Fields*** each one having a name, a type and possibly an initial value. Data fields can be referenced wherever in the process e.g. in the conditions associated to the transitions, as actual parameters for application and subflow invocations and in the pieces of code triggered by code activities.

Figure 5 shows an example summarizing the main elements of the WADE meta-model.

5.2 Process elements implementation

Having presented the main elements that make up a workflow process, in this section we show the structure of the Java code that actually implements them. It is important to note that, even if they could, WADE developers are not required to write the pieces of code that are described in this section directly. The graphical editor provided by Wolf automatically manages them all.

5.2.1 Structure of a workflow class

Each workflow process is implemented by a Java class that extends (directly or indirectly) the `WorkflowBehaviour` base class.

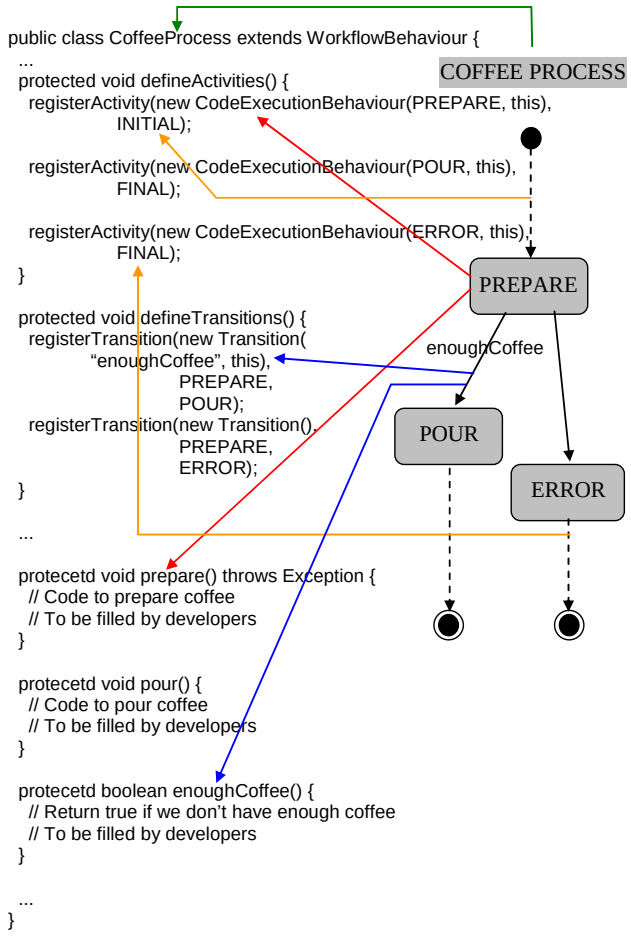

Figure 6. Workflow class structure

The `WorkflowBehaviour` class on its turn extends the JADE `FSMBehaviour` class and provides on top of it an API consistent with the meta-model presented in 5.1. In particular the `registerActivity()` and `registerTransition()` methods allow defining the activities and transitions that specify the process flow. Even if in principle this is not strictly necessary it is highly recommended to register all process activities in a method called `defineActivities()` and all transitions in a method called `defineTransitions()`. This is because the workflow graphical editor included in Wolf searches for these methods to detect process activities and transitions to show.

The `registerActivity()` method gets the behaviour implementing the registered activity as an argument. More in details there is a class for each type of activity: `ToolExecutionBehaviour` to register a tool activity, `SubflowDelegationBehaviour` to register a subflow activity, `CodeExecutionBehaviour` to register a code activity and so on. The actual operations to be performed in a registered activity (no matter of its type) are specified in a `void` method of the workflow class. That method must have the same name as the corresponding activity. Each activity behaviour is just responsible for invoking the related method when the activity is visited.

Similarly the `registerTransition()` method gets a `Transition` object as an argument. In case the transition has an associated condition, this is implemented by a `boolean` method of the workflow class. The Transition object is just responsible for invoking that method when the transition condition must be evaluated.

Figure 6 summarizes the above correspondences by showing the structure of a workflow class implementing a simple process for coffee preparation.

5.2.2 Data fields and formal parameters

Data Fields of a workflow process are implemented as fields of the workflow class. For instance, with reference to the coffee process shown in Figure 5, there could be a data field containing the amount of coffee in grams available for preparation. This would be implemented as an `int` field of the `CoffeProcess` class.

Workflow formal parameters need to be accessed by the workflow in a similar way to data fields. Therefore they are implemented, like data fields, as fields of the workflow class. In order to let WADE know that they are formal parameters, however, they must be annotated by means of the `FormalParameter` annotation. For instance the coffee process could take an input formal parameter `nCups` indicating the number of cups to be prepared. This would be implemented as an `int` field annotated as below.

```
@FormalParameter(mode=FormalParameter.INPUT)
private int nCups;
```

5.2.3 Layout information

When representing a process as a workflow, besides the information related to the actual flow of execution, it is necessary to consider all additional information such as activity positions, transition bend-points (if any), comments and so on. In WADE all these information are captured in the @WorkflowLayout annotation of the workflow class. By means of this choice the execution flow definition remains clean and readable as much as possible and at the same time WADE deals with a single java file per workflow.

5.3 Delegations

An important characteristic of WADE micro workflow engine is the possibility of delegating subflows to other agents selected at runtime and possibly running on remote hosts. Subflow performer selection criteria are clearly application specific. For instance they can be based on the current host/agent workload thus achieving a GRID-like system able to distribute pieces of a complex process across available hosts. In the Network Neutral Element Manager system described in chapter 2, this feature was deeply exploited to deal with agent specialization. As an example the IPTVServiceActivation workflow, that is triggered when network elements must be configured to support a new ADSL IPTV service, is cooperatively executed by at least three different agents. It is initially submitted to a ServiceAgent embedding additional features to interact with the ADSL services DB. The ServiceAgent executes the part of the process that stores the new service in the services DB and then delegates the actual network element configuration part to a TopologyAgent. The latter, as its name suggests, knows the topology of the network and identifies, on the basis of the customer location and on the type of the service to be activated, the network elements involved in the activation process. Having done that, the TopologyAgent delegates the specific configurations to the ResourceProxy agents acting as proxies for the identified network elements.

5.4 Workflow Inheritance

One of the requirements that were taken into account when designing the WADE micro workflow engine was the support for workflow inheritance. This feature, that allows creating new workflows starting from existing ones and specifying only the differences, is likely one of the most characterizing in the landscape of workflow management tools.

```
public class ExtendedCoffeeProcess extends CoffeeProcess {
  ...
  protected void defineActivities() {
    super.defineActivities();
    deregisterActivity(ERROR);

    registerActivity(new CodeExecutionBehaviour(BUY, this));
  }

  protected void defineTransitions() {
    deregisterTransition(PREPARE, ERROR);

    registerTransition(new Transition(),
                       PREPARE,
                       BUY);
    registerTransition(new Transition(),
                       BUY,
                       PREPARE);
  }

  ...

  protecetd void buy() throws Exception {
    // Code to buy coffee
    // To be filled by developers
  }
  ...
}
```

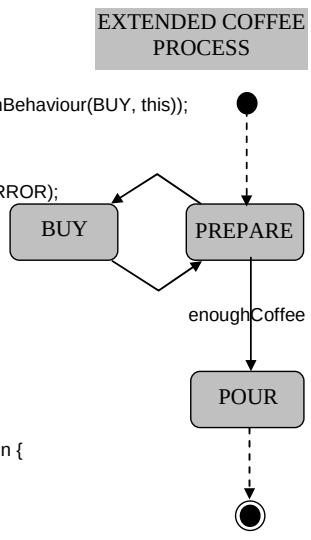

Figure 7. Workflow inheritance example

More in details, being WADE workflows Java classes, they can be extended using Java inheritance. This allows redefining the methods associated to the process activities thus modifying the related operations according to the needs of the extended workflow. Furthermore the WorkflowBehaviour class provides the deregisterActivity() and deregisterTransition() methods that, in conjunction with the registerActivity() and registerTransition() methods described in section 5.2.1, allows modifying the process execution flow at will. For instance we could create an extended version of the CoffeeWorkflow presented in section 5.2.1 as shown in Figure 7.

6. CONCLUSIONS

In this paper we presented two mission critical applications developed in Telecom Italia and currently deployed in the field. These applications have a direct influence on the work of thousands of technicians and on millions of customers and, as a consequence, have strong requirements in terms of scalability and flexibility. The enabler for this compelling price/performance proposition is the middleware 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 tolerance environment that runs on a grid of low cost servers
- 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 to the level of system internal logics, WADE provides a workflow view over normal Java classes. This allows combining the flexibility of the Java language and the power of the Eclipse IDE with the expressiveness and traceability of the workflow metaphor.

On the other hand the exploitation of “autonomy” and “self consciousness ” of agents (and in general of Artificial Intelligence Techniques) still encounters some resistance especially in big companies that need deep control over their systems. The NNEM and Wizard applications described in this paper currently do not take advantage from these features.

Future activities are focusing on empowering the WADE platform and its development environment Wolf. In particular a deep integration with Web services and the support for asynchronous events are in the roadmap.

7. ACKNOWLEDGMENTS

We gratefully acknowledge all of our international friends who have contributed to Jade over the years. Furthermore we are grateful to our Telecom Italia colleagues (in primis the OSS Innovation department and the Trento Software factory) for the contributions to the systems implementation, deployment, testing and maintenance of WADE, NNEM e Wizard

8. REFERENCES

[1] The individual shareholder Guide Oct. 2007. http://ticlub.telecomitalia.com/pdf/Guida_1H_2007_ottobre_EN.pdf

[2] Bellifemine F., Caire G., D. Greenwood. Feb. 2007, Developing multi-agent systems with JADE. Wiley Series in Agent Technology. ISBN 978-0-470-05747-6,.

[3] Bellifemine F., Poggi A., Rimassa G.. 2001. Developing multi agent systems with a FIPA-compliant agent framework. In Software - Practice & Experience, 31:103-128,.

[4] Berger M. Rusitschka S., Toropov D.,Watzke M., Schlichte M., 2002. Porting Distributed Agent-Middleware to Small Mobile Devices. In Workshop on Ubiquitous Agents, AAMAS 2002.

[5] JADE - Java Agent Development framework. http://jade.tilab.com.

[6] Lee H., Mihailescu P., Shepherdson J., . 2007, Realising Team-Working in the Field: An Agent-based Approach, IEEE Pervasive Computing, pp. 85-92 .

[7] AgentLink III. Agent Technology Roadmap. Available from http://www.agentlink.org/roadmap/index.html

[8] Belecheanu R., Munroe S., Luck M., Payne T., Miller T., Pechoycek M., McBurney P, 2006 . Commercial applications of agents lessons, experiences and challenges, Industrial Track Fifth international joint Conference on Autonomous Agents and Multi-Agents Systems. ACM Press, pp 1549-1555

[9] Buhler P.A., Vidal, J.M. 2005, Towards Adaptive Workflow Enactment Using Multiagent Systems. Information Technology and Management, Information Technology and Management, 6(1):61-87.

[10] Poggi A., Tomaiuolo M., Turci P, 2007, An Agent-Based Service Oriented Architecture http://woa07.disi.unige.it/papers/PoggiSOA.pdf

[11] Foster I., Jennings N. R., Kesselman C., 2004, Brain Meets Brawn: Why Grid and Agents Need Each Other, Proc. Autonomous Agents and Multi Agent Systems (AAMAS 2004), pp 8-15

[12] Greenwood, D., Callisti, M. Engineering Web Service-Agent Integration. In IEEE Conference of Systems, Man and Cybernetics, 2004. Available from http://www.whitestein.com/resources/papers/ieeesmc04.pdf

[13] Savarimuthu, B.T.R.; Purvis, Mary.; Purvis, Mart.; Cranefield, S., 2005.,Integrating Web services with agent based workflow management system (WfMS), Proceedings. The 2005 IEEE/WIC/ACM International Conference on Web Intelligence, Page(s): 471 - 474

[14] Negri A., Poggi A., Tomaiuolo M., Turci P,. 2006, Dynamic Grid Tasks Composition and Distribution through Agents,. Concurrency and Computation: Practice and Experience(2006), 18(8): 875-885

[15] Eclipse. www.eclipse.org.

[16] Gotta,D, Covino, G., 2004, "GRID COMPUTING in the real world", CMG Italia, Italy conference Pisa 2004, http://www.cmgitalia.it

[17] L.A. Barroso, J. Dean, U. Holzle, 2003 "Web search for a Planet: the google cluster architecture", Micro, IEEE,Volume: 23, Issue: 2, Page(s): 22- 28

[18] WFMC WorkFlow Managment Coalition, http://www.wfmc.org/

[19] XPDL XML Process Definition Language, http://www.wfmc.org/standards/xpdl.htm

[20] van der Aalst W.M.P.,. 2003,Patterns and XPDL: A Critical Evaluation of the XML Process Definition Language. QUT Technical report, FIT-TR-2003-06, Queensland University of Technology, Brisbane,

[21] Shapiro, R. 2002. A comparison of XPDL, BPML and BPEL4WS (Rough Draft), Cape Vision

[22] BPMN Business Process Modeling Notation http://www.bpmn.org/

[23] P. Wohed, W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede., 2003 Analysis of Web Services Composition Languages: The Case of BPEL4WS., 22nd International Conference on Conceptual Modeling (ER 2003), volume 2813 of Lecture Notes in Computer Science, pages 200-215. Springer-Verlag, Berlin, 2003.

[24] WS BPEL Web Services Business Process Execution Language Version 2.0, OASIS Standard, 2007, Available from http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf

[25] Tele Management Forum. www.tmforum.org

[26] G. Covino, D.Gotta, A.Nasuto, 2007. La gestione delle nuove reti con un diverso paradigma, pp 11-13. http://www.telecomitalia.it/TIPortale/docs/innovazione/012007/Pag27_42_NNEM.pdf