Enacting BPM-oriented Workflows with Wade

Federico Bergenti
Dipartimento di Matematica
Università degli Studi di Parma
43100, Parma, Italy
federico.bergenti@unipr.it

Giovanni Caire, Danilo Gotta,
Daniela Long, Giovanna Sacchi
Telecom Italia S.p.A.
10148, Torino, Italy
{giovanni.caire,danilo.gotta,daniela.long,giovanna.sacchi}@telecomitalia.it

***Abstract*—This paper has two main focuses. First it provides a review on the reasons why agent technologies are a good choice for BPM (Business Process Management). A brief survey of the literature on the subject is presented and a critical revision of the main motivations that are commonly accepted for the use of agents in BPM is presented taking into account recent technological developments. Then, the paper presents the recent developments of Wade (Workflow and Agent Development Environment) and it confers such developments and value-added features in the scope of the initial discussion. Finally, the paper briefly enumerates some successful applications of the presented technologies in Telecom Italia. Such applications are so important and demanding that their implementation using agent-based approaches is an outstanding result for agent technology.**

Keywords-business process management; agent-based business process management systems; Wade

I. Introduction

Business Process Management (BPM) is now a consolidated trend in IT that has recently came up as a new discipline intended to unify related topics such as Process Modeling, Workflows, Enterprise Application Integration and Business-to-Business integration (see, e.g., [8]).

Despite the complexity of the subject, we can broadly refer to a *business process* as a set of interdependent activities that collectively realize a business objective or policy, within the context of an organizational structure defining the functional roles and the relationships between actors [15]. BPM includes the following activities regarding business processes [11]:

1. Process description: every process must be described in some specification language in order to enumerate the activities that need to be performed, the actors who perform them, and the interdependencies that exist between activities; and
2. Process execution and management: organizations typically use a software system, called *BPM system*, in charge of enacting the process description and turn it into practice.

While the importance of BPM systems in process execution is obvious, it is of equal importance to couple BPM systems with models intended to express the complexities of business processes in the scope of their organizational context, and to support reasoning about processes for enabling future optimization and reengineering activities.

Generally speaking, a BPM system enables a wide range of tasks like automating manual work, improving information and knowledge exchange among employees, controlling business processes in place, and assist in design and engineering of business processes. More in details, there are a few features that every BPM system must provide and that we consider of paramount importance (see also [8]):

1. It should transparently support multiple instances of a given process and a given task;
2. It should ensure that dependencies between the tasks are timely satisfied;
3. It should allow user activities to be assigned appropriately; and
4. It should integrate with the enterprise software tools required to complete the tasks.

The introduction of a software system for BPM typically entails the adoption of appropriate workflows within the enterprise. A *workflow*, as defined in [15], is the automation of a business process–in whole or part–during which artifacts, information and/or tasks are passed from one actor to another according to a set of procedural rules. Normally, workflows are meant to ensure that the right people receive the right information at the right time.

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., [3][15]). 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 lead 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][13]). Therefore, 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. The promise of agent technologies with this respect is to provide solid warranties for greater dynamism, agility, and adaptability.

We already have a number of agent-based BPM systems available (see, e.g., [4][6][11][12][13]) and all such proposals share the common factor of using the autonomous and collaborative nature of agents to accommodate unexpected situations in dynamic business processes.

II. Agents and Business Process Management

In order to precisely discuss the role of agent technology in the scope of BPM systems, we must first review in details what a BPM system is and how it is expected to behave. The most relevant reference for this kind of systems is [15], which characterizes a BPM system as a software system that defines, creates and manages the execution of workflows that are running on one or more workflow engines. Such workflow engines are able to interpret the process definition, interact with workflow participants and, where required, invoke the use of other software.

Such BPM systems are typically modularized in a set of well-defined parts (see, e.g., [8]): business process definition tools, business process servers, business process client applications and business process monitoring and administration tools. Figure 1 provides a pictorial view of such a modularization of a BPM system.

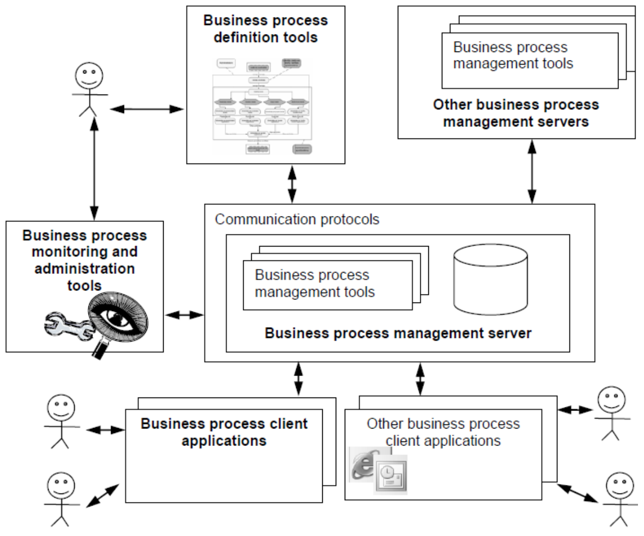
Business process definition tools
Business process management tools
Other business process management servers
Communication protocols
Business process monitoring and administration tools
Business process management tools
Business process management server
Business process client applications
Other business process client applications

Figure 1. Conceptual model of traditional BPM system (from [13])

Business process definition tools allow modeling the process in terms of workflows, actors, tasks, activities and their relationships and interdependencies. This is normally done using a graphical notation that typically resembles flowcharts.

Business process servers are the software systems that provide the runtime execution of defined processes. They read process definitions and actually execute and track them.

Business process client applications are software systems that actors use to interact with the workflow. The application does not need to be part of the BPM system and it is typically a thin (Web) client that behaves as a front end to allow users receiving information and submitting events to the business process server.

Business process monitoring and administration tools are intended to provide a real-time view of the state of execution of workflows and they provide means to manage unforeseen situations. They are valuable tools that give concrete help at runtime and that trace the information needed to optimization and reengineer processes.

Even if the modularization of typical BPM systems is well established and understood, in principle different systems can have different approaches to support the lifecycle of business processes. Unfortunately, according to [15] the majority of current generation BPM system shares a common approach to structure the lifecycle of business processes. They all start modeling business process from activity analysis and they pay primary attention to business process tasks interdependences in order to correctly enact known sequence of the tasks [8]. All in all, such systems are adequate only in situations where a business process is fully understood and every conceivable outcome of tasks and activities can be considered and controlled beforehand.

As we briefly discussed before, not all business processes can be defined with such a fine level of control at design time. Real-world business processes are complex and continuously changing in order to accommodate the changes of their operative environment. Because of that, [8] provides a list of the major drawbacks and limitations of current BPM systems, which we review here according to recent developments of the technology:

1. Limited flexibility during process enactment;
2. Inability to cope with dynamic changes in the availability of resources needed to accomplish activities and tasks, as existing systems tend to lack the necessary facilities to redistribute work items automatically as (and when) required;
3. Inadequate handling of exceptional situations, especially when an exceptional case arises in a part of compound (yet possibly recoverable) tasks;
4. Limited (or even, no) ability to predict changes due to external events, in both the volume and the time distribution of activities.
5. Insufficient interoperability with other systems as the majority of existing BPM systems consists of centralized and monolithic systems that are meant to control their operative environment and that are not designed to cooperate with other (possibly unknown) controllers.

Even a superficial read of the mentioned drawbacks suggests that agent technology should be capable of addressing and effectively solving all such issues. If agent technology is involved in the enactment of business processes, we should benefit from the intrinsic dynamism and flexibility of agent-based systems.

An agent-based BPM system is made of a set of software modules that meet the coarse grained criteria that define *agenthood* and that are involved in managing the flow of work throughout a business process [13]. The basic idea is to rethink the mentioned modules of a traditional BPM system in terms of interacting agents in charge of peculiar responsibilities and capable of predicting and reacting to unforeseen situations. This does not mean that we need to rethink the discussed modularization of a BPM system; rather agents give us the possibility of going deeper in the definition of the parts of a BPM system. All such parts are then viewed as agents in order to benefit from the intrinsic characteristics of agents themselves.

Moreover, the use of agents enables another, orthogonal, modularization possibility, as suggested in [8]. An agent-based

BPM system can split a business process into parts and trust the control over such parts to individual agents.

Finally, the business logic behind the business processes can be explicitly defined to agent (e.g., by means of some set of business rules), to allow agents reasoning on their and others' roles in a business process. Agents use business logic to plan their activities in order to achieve their goals and to meet the overall goals of the business logic.

Given such a view of an agent-based BPM system we can sum up the major advantages of such an approach to building BPM systems [3][11]:

1. The use of goal oriented, communicating autonomous agents, which although concerns about business logic, allows multiple solution paths to the business process goals to be achieved;
2. The agent metaphor allows decentralized ownership of the tasks, information and the resources involved in business processes;
3. The use of agents provides high degree of natural concurrency, when many interrelated tasks are running at any given point of the business process;
4. The decoupling of the parts of the system that agent technologies ensure allows them to be swapped out, replaced, or even added to the system without impacting other parts; and
5. Agent-based technologies allow building highly decentralized, distributed systems, which corresponds to the real-world situation, when the business processes in organizations are physically distributed.

Unfortunately the literature has already identified some disadvantages of promising agent-based approach to BPM systems (see, e.g., [11]). We summarize the most prominent here for the sake of completeness:

1. The agent-based systems have no overall system controller, which implies that the agent-based approach is not the best choice for managing business processes with a lot of global constraints to be satisfied; and
2. Agent-based systems have no global complete knowledge, i.e., an agent's actions are–by definition–determined by that agent's local knowledge and this may mean that agents could make globally sub-optimal decisions.

It is worth noting that such issues are actually common to all agent-based software systems and they are not typical of BPM systems. Actually, such issues and their importance originate from the common understanding of agent-based systems as useful *only* in a limited set of environments that are characterized by intrinsic dynamism and uncertainty. Obviously not all operative environments are so critical and we believe that agent technology can work effectively also in more traditional settings. The work presented in this paper is precisely motivated by such a point of view: we think that agent technology is now ready to deliver very solid, scalable and visually programmable software systems even in traditional environments where dynamism and uncertainty are not major issues.

We have been using agent technology in traditional operative environments for its maturity and effectiveness in the provision of nonfunctional features coupled with the possibility of visually programming complex behaviors. Next section presents the Wade (Workflow and Agent Development Environment) [6], which is an agent-based BPM system that has been successfully adopted in a number of mission critical software systems–as detailed further at the end of the paper–for the possibilities it provides in the realization of solutions with distinguished nonfunctional requirements in terms of scalability and robustness. The role of Wade-based agents in such systems is not about using the autonomy of agents in the management of dynamic and unforeseen situations; rather it is about providing developers with friendly tools that provide a robust shield against the complexity of nonfunctional requirements.

Finally, it is worth noting that the tight integration of Wade with mainstream development technologies like Java and Web Services allows developers incrementally adopting agent technology in their systems. The parts of the system that can fruitfully empower the features of agents are easily developed using Wade and related tools (e.g., Wolf [6] and Jade [10]), while other parts are still developed using mainstream technology with no effort needed for integration.

III. WADE: AN AGENT BASED FRAMEWORK LEVERAGING THE WORKKFLOW METAPHOR

Wade (Workflow and Agent Development Environment) [6] is an open source framework meant to develop distributed applications, based on the agent paradigm and the workflow metaphor. It is built on top of Jade [10], the popular open source middleware for the implementations of multi-agents systems, complying with FIPA specifications [7].Wade adds to Jade the possibility to define tasks in terms of workflows and a set of mechanisms to handle the complexity of administration and fault tolerance operations in a distributed environment.

Wade was initially conceived to exploit the workflow approach in the implementations of the system internal logics that can be modeled in terms of "short running" processes. Such kind of processes are generally characterized by a short execution time (typically seconds or in some cases minutes) and a high CPU time consumption and can be defined in terms of the activities to be executed, the relations between them (used to specify the execution flow) and the conditions of start and termination.

Many advantages have been demonstrated to become effective following this approach and, among them, it is worth mentioning the possibility to have a graphical representation of a workflow, easily understandable by domain experts as well as by programmers. Because of the workflows expressiveness, domain experts can directly validate the system logics and, in some cases, they could even contribute to the actual development of the system with no need of programming skills.

Consistently with the aforementioned requirements regarding short-running processes, some design decisions have been taken. First, workflows are modeled in terms of Java code to ensure maximum efficiency and flexibility. In the literature several formalisms, such as XPDL, BPEL, WS-BPEL (e.g., see [14][17]) can be found to describe workflows. However, if on the one hand they provide a clear and intuitive representation of the process execution flow, on the other hand they are not suitable to specify all the details involved in the implementation of a piece of the business logic of a given software system. A common programming language like Java is definitely more powerful and flexible to deal with data transformations, computations and other low level auxiliary operations that are often needed when specifying the business logic of the system under development.

Taking into account the above consideration, Wolf (WOrkflow LiFe cycle management environment), the Wade graphical editor, provides a workflow view on top of a Java class with a well defined structure (see also [1][2] for similar graphical languages for agent-based workflows). Wolf has been developed as an Eclipse plug-in, thus allowing the exploitation of all features offered by the Eclipse Java IDE.

Finally, because workflows start and terminate their executions in a short time, no persistency mechanism has been considered necessary and workflows did not survive to the shut-down of their Wade platform.

Summing up, until version 2.6 of Wade, the main target of Wade was the implementation of the system internal logics, using the workflow metaphor and a key element of this approach was the choice to model a workflow directly by means of a Java class, providing a graphical representation of it using Wolf.

IV. BPM-Oriented Evolution of Wade

Starting from 2010 new requirements coming from Telecom Italia Wade-based systems as well as the Open Source Community shown that, though very effective for a certain type of applications, the followed approach restricted too tightly the actual usages of Wade. In particular, more and more frequently the need to properly manage situations where a workflow could block waiting for external events that may happen in hours, days or even months was indicated as a mandatory feature.

To meet such ever growing requirements with version 3.0, Wade had a strong evolution that, though preserving its distinctive characteristics, makes it now a tool that can effectively play the role of orchestrator in a BPM context.

A. Long-Running workflows

The base for all Wade BPM-oriented features described in this section is the possibility of having workflows that survive to a system restart. Such workflows are identified as *long-running*. More in details, if the platform is shut down while a long-running workflow W has executed activity A_n, as soon as the platform starts up again workflow W is automatically reloaded and forced to recover its execution starting from activity A_{n+1}. Under the hood Wade saves the status of a long-running workflow on a persistent storage after the execution of each activity. The persistent storage is implemented by a relational database accessed through Hibernate. The mechanism has been tested with a number of different database management systems, e.g., H2, mySql and Oracle. A new administrator agent called WSMA (Workflow Status Manager Agent) has been introduced and it is responsible to manage all operations related to tracing, persisting and recovering the status of workflows.

B. Asynchronous events

Another major step-forward in the evolution of Wade is the introduction of an integrated event sub-system implemented as an agent called ESA (Event System Agent). When developing a workflow, besides regular activities, it is now possible to include new synchronization activities that, when reached, make the execution block until a given event happens. More in details, when the process enters such a synchronization activity, the workflow thread is released (to prevent resource consumption) and the WSMA switches the workflow state from ACTIVE to SUSPENDED. A suitable API (the EventChannel API) is provided to submit events to the event system. As soon as an event matching the template specified in the synchronization activity is submitted, the workflow is resumed (a new thread is allocated to it) and its state is switched back to ACTIVE. Furthermore, any information carried by the event is made available to the workflow for further processing. The event system stores received events for a configurable amount of time so that it is now possible to transparently deal with situations where a synchronization activity is reached after the expected event happened. In such cases the workflow does not even block and immediately moves forward. It should be noted that the possibility of blocking to receive asynchronous events is not strictly related to long-running however, if the system is restarted, while long-running workflows will be recovered transparently, all suspended short-running workflows will be immediately aborted.

C. Web Service exposure

Since version 2.0 Wade includes a powerful embedded support to invoke Web services from within a workflow. In version 3.0 such a support is enriched with the possibility of automatically exposing Web services. This feature is twofold. On the one hand it is possible to expose the operations specified in a given WSDL and block a workflow waiting for a given operation to be invoked. This is achieved by combining the new Web service exposition feature with the support for asynchronous event described in previous section. An ad hoc *WaitWebService* synchronization activity now exists that, when reached, makes the workflow block until the event corresponding to the invocation of a previously exposed Web service operation happens. Internally the code serving the Web service invocation encapsulates the operation parameters into an event and submits it to the event system. On the other hand, it is now possible to automatically expose a workflow as a Web service. The workflow name maps to the service name and a single *execute* operation is generated with parameters matching workflow's ones. The code serving the invocation triggers the execution of the workflow. This feature is made available in Wolf by means of a simple click on the workflow class.

From the architectural point of view, the Web service exposition feature described in this section is implemented by a new component called *WadeServices*. This is a standard Web application that can be executed within any servlet container such as Apache Tomcat.

D. *Administration Web Console*

According to the new evolutions of Wade and in order to facilitate the administration of the platform, a Web console has been developed to allow performing both low level management operations, like the start-up/shut-down of the platform, and high level actions, more related to the business logics, like browsing and launching a workflow.

This Web console has been implemented using the ZK [18] framework, an open source solution to develop Web applications, based on AJAX technology. In particular, the ZK framework has been extended to support new ZK components specially intended to support the Wade administration functionalities. Such components, exploited by the Web console, can be also reused by developers inside of custom Web applications that need to integrate Wade platform management functions.

V. CONCLUSIONS

This paper looks through general questions about agent-based BPM. It gives an overview of the main concepts of BPM and it identifies a general conceptual model of centralized and agent-based BPM systems. Then, the paper points out key properties of agent-based BPM systems, and it sums up main advantages and disadvantages of such systems. It is worth noting that this paper does not pay much attention to implementation issues inherent to the introduction of agent technologies.

The existing agent-based BPM systems, that have already been developed and applied as the solution of real world problems, proves that agent technologies are a highly perspective direction for future researches in this field. From our experience we now think that the main issues which the designer of an agent-based BPM system should be aware of are: inter-agent communication protocols, agent action planning (which is itself a topic for future researches), business logic representation.

From a methodological point of view, Wade has been appreciated in the development of mission critical agent-based BPM systems for the agile approach that it brings in. Wade and related tools provide a solid platform for the development of complex BPM systems that tightly integrate the power of a visual approach with scalability, robustness and interoperability with mainstream technologies. This has reduced the effort needed to develop effective demonstrators and prototypes that were fruitfully scaled up to the core parts of real systems, thus reducing time-to-market and improving the overall qualities of the systems and of the development processes.

In particular, Wade proved to be largely useful to develop single applications and service oriented architectures with strong requirements regarding performances, scalability and high flexibility in defining the systems' logics.

In Telecom Italia Wade is used for a number of mission critical systems [6][15] that are now in everyday use with real users and in the scope of projects with real customers:

- NNEM implements a mediation layer between network elements and OSS systems for millions of Telecom Italia customers;
- Wizard provides step-by-step guidance to thousands Telecom Italia technicians performing installation and maintenance operations in the fields with more than 1 million documented assisted installation since 2007; and
- WeMash, a mashup platform for service-oriented architectures whose target is to enable non-developer users to self-create simple applications and to share them within the team they are working in.

The results were so compelling that Telecom Italia chose Wade as the enabling middleware for a *Software As A Service* (SAAS) offer for Utilities customers in the fields of electricity, gas and water. This offer includes various systems (Wizard 2.0, WeMash, and a bus orchestrator) based on the new functionalities of WADE 3.0 described in this paper with a fully functional service oriented architecture based completely on open source components.

REFERENCES

[1] Bartocci, E., Corradini, F., Merelli, E. (2006) Building a MultiAgent System from a User Workflow Specification. In *Procs. Workshop "Dagli Oggetti agli Agenti"*.

[2] Bartocci, E., Corradini, F., Merelli, E., Scortichini, L. (2007) BioWMS: A Web-based Workflow Management System for Bioinformatics. *BMC Bioinformatics* 8(S-1).

[3] Bolcer, G. A., and Taylor, R. N. (1998) Advanced Workflow Management Technologies, *Software Process: Improvement and Practice*, 4(3):125–171.

[4] BPMN – Business Process Modeling Notation, available at http://www.bpmn.org/

[5] Cai, T., Gloor, P. A., and Nog, S. (1996) *DartFlow: A Workflow Management System on the Web Using Transportable Agents*. Dartmouth College.

[6] Caire, G., Gotta, D., and Banzi, M. (2008). WADE: a software platform to develop mission critical applications exploiting agents and workflows. In *Procs 7th Int'l Conf Autonomous Agents and Multiagent Systems*, 29-36.

[7] FIPA – Foundation for Intelligent Physical Agents. Available at http://www.fipa.org

[8] Grundspenkis, J., and Pozdnyakov D. (2006) An Overview of the Agent Based Systems for the Business Process Management. In *Procs Int'l Conf Computer Systems and Technologies*.

[9] Hawryszkiewycz, I., Debenham, J. (1998) A workflow system based on agents. In *Procs 9th Int'l Conf Database and Expert Systems Applications*.

[10] JADE – Java Agent Development framework. Available at http://jade.tilab.com

[11] Jennings, N. R., Faratin, P., Johnson, M. J., Norman, T. J., and Wiegand, M. E. (1996) Agent–Based Business Process Management.

[12] Jin, W., and Chang, S. T. (1996) Agent-based Workflow: TRP Support Environment (TSE). *Computer Networks and ISDN Systems*, 28(7–11):1501-1511.

[13] Pang, G. (2000) *Implementation of an Agent-Based Business Process*. University of Zurich.

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

[15] Trione L., Long D., Gotta D., and Sacchi G. (2009) Wizard, WeMash, WADE: Unleash the Power of power of collective intelligence. In *Procs 8th Int'l Conf Autonomous Agents and Multiagent Systems*.

[16] Workflow Management Coalition. Workflow Management Coalition Terminology & Glossary, available at http://www.wfmc.org

[17] Workflow Management Coalition. XPDL XML Process Definition Language, available at http://www.wfmc.org

[18] ZK – Available at http://www.zkoss.org/