Latest Developments of WADE to Support User-Centric Business Processes

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

[1] Università degli Studi di Parma, 43124, Parma, Italy
federico.bergenti@unipr.it
[2] Telecom Italia S.p.A., 10148, Torino, Italy
{giovanni.caire,danilo.gotta}@telecomitalia.it

Abstract. In this paper we present the latest developments of *WADE* (*Workflows and Agents Development Environment*) that provide concrete support for a better realization of the innovative paradigm of *agent-based BPM* (*Business Process Management*). First, we review and critique the basic ideas behind agent-based BPM and we focus on its innovative characteristics with respect to traditional BPM. Then, we describe the most recent developments of WADE that are intended to enhance its agent-based runtime platform by providing improved non-functional features and a better integration with the external software systems. Finally, we discuss the new functionality that WADE offers to enable the rapid and effective realization of *user-centric business processes*, i.e., business processes that are tightly integrated with the work of users and that are mainly driven by user interactions. Such processes are met frequently in practice and WADE seamlessly accommodates Web and Android users by means of dedicated views. We conclude this paper with a brief overview of notable mission-critical applications that are already using WADE and its new features.

Keywords: Agent-based BPM, user-centric business processes, WADE.

1 Introduction

Business Process Management (*BPM*) 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 (see, e.g., [12]). BPM is today considered essential to the life of complex and dynamic enterprises and the research on the subject from IT and other perspectives is very active.

Despite the complexity of the subject that has been promoting interesting and longstanding debates, 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 that defines the functional roles and the relationships between actors [22]. With this respect, BPM includes at least the following activities regarding business processes [15]:

M. Baldoni et al. (Eds.): DALT 2012, LNAI 7784, pp. 20–36, 2013.

- Process description: every process must be described in some specification language in order to enumerate *(i)* the activities that need to be performed, *(ii)* the actors that perform them, and *(iii)* the interdependencies and related constraints that exist between activities; and
- Process execution and operational management: organizations typically use a software system, called BPM system, which is in charge of enacting the process description and turn it into practice.

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 [12]):

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

While the importance of BPM systems in process execution is obvious, it is of equal importance to couple BPM systems with the 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. It is often the case in the practice of BPM that the approach proposed by the adopted BPM system becomes the driving force of BPM and we often see business processes accustomed to systems and not vice versa. Such a counterintuitive approach is quite common practice in small and dynamic enterprises and this is the main reason why we believe that the role of the BPM systems in the large scope of BPM is often underestimated.

The introduction of a BPM system typically entails the adoption of appropriate workflows within the enterprise. A workflow, as defined in [22], 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.

Among the large variety of possible classification of business processes, it is worth noting that even the general definition of workflow that [22] proposes stresses the central role of actors in BPM. Then, if we match such a general definition with the everyday practice of BPM, we note that normally workflows are designed to ensure that the right people receive the right information at the right time. Workflows are often used to guide the work of people and we often witness workflows that are only meant to interact with the users of the BPM system. Such users are central subjects of BPM, exactly like BPM systems, and we strongly believe that they deserve a special treatment on their own.

In order to emphasize the role of users in the very general landscape of BPM and to justify the very frequent use of business processes to drive the work

of people, we talk about *user-centric business processes* every time a business process is primarily intended to interact with the users of the BPM system in order to guide and provide assistance on his or her operative work.

User-centric business processes are so frequent in practice and their importance is so relevant for actual uses that modern BPM systems are requested to reserve a special treatment to them in order to enable specific functionality and promote optimizations. Unfortunately, little or no attention to such a kind of processes is paid by traditional BPM systems and interaction with users is often underestimated as yet another type of event. Such an approach is obviously very generic, but it misses a relevant part of the peculiarity of such processes, i.e., the need for the system to provide a means for users to effectively interact with the business processes. In the description of a user-centric business process we need to precisely describe the way users are presented the relevant information regarding the state of the process, i.e., the view that the user has on the process. Moreover, we also need to precisely describe the information that the user is expected to provide in order for the process to continue smoothly. Both such characteristics are of remarkable importance in practice and they deserve special care that traditional BPM systems do not always provide. Traditional 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., [4]). 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., [15,17]). Therefore, we witness the rapid evolution of alternative approaches to traditional BPM that notably include *agent-based BPM*, and more generally, the adoption 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., [1,2,6,7,13,16,17]) 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. This is a characteristic feature of agent-based BPM systems that is often used to motivate their adoption, but it is worth noting that agent technologies today provide so crucial advantages (e.g., in terms of non-functional features at runtime and effectiveness at design time) that their use should not be limited to the situations in which dynamism is a critical requirement.

In the practice of using agent-based BPM systems in real-world contexts (see, e.g., the concluding section and [7,21]) we noticed that users often appreciate them for their ease of use or for their robustness and dependability, rather than for their ability to cope with dynamic and unexpected situations, which are very rare and of marginal interest. The rest of this paper focuses on the weaknesses of traditional BPM systems that we have just emphasized and it presents a set of tools that effectively address them.

In the following section we describe the view of agent-based BPM that *WADE* (*Workflows and Agents Development Environment*) [7] promotes and we sketch a comparison with more traditional solutions. Then, Section 3 presents the most recent developments of WADE that have turned it into a full-featured agent-based BPM platform with improved non-functional characteristics and better interfaces with external software systems. All described features of WADE are available since version 3.1 (dated July 2012), which also includes a specific support for user-centric business processes, as described in Section 5. Finally, we close the paper with a brief summary of mission-critical systems based on WADE that are in everyday use in Telecom Italia.

2 Agent-Based BPM

In order to properly discuss the role of agent technologies in the scope of BPM, 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 [22], 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., [12]) as follows:

- Business process definition tools: they 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: they are the software systems that provide the runtime execution of defined processes. They read process definitions and actually execute and track them.
- Business process clients: they 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 to receive information and to submit events to the business process server.
- Business process monitoring and administration tools: they 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 [22], the majority of current generation BPM systems share a common approach to structure the lifecycle of business processes. They all start modeling business process from activity analysis and they

pay principal attention to business process tasks interdependences in order to correctly enact known sequence of the tasks [12]. 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, [12] provides a list of the major drawbacks and limitations of traditional BPM systems, which we review here taking into account recent developments in the field:

- Limited flexibility during process enactment;
- 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;
- Inadequate handling of exceptional situations, especially when an exceptional case arises in a part of compound, yet possibly recoverable, tasks;
- Limited or even null ability to predict changes due to external events, in both the volume and the time distribution of activities; and
- Insufficient interoperability with other software systems, as the majority of existing BPM systems consist 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 technologies are capable of addressing and effectively solving all such issues. If agent technologies are involved in the enactment of business processes, we benefit from the intrinsic dynamism and flexibility of agent-based systems and we rely on mature technologies that provide solid solutions to common software development issues. Moreover, the use of agent technologies can fruitfully enable a *declarative* approach to BPM that has already gained a significant interest for its inherent characteristics (see, e.g., [18,20]).

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 [15,17]. 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 characterization 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 [12]. An agent-based BPM system can split a business process into parts and trust the control over such parts to individual agents.

Given such a view of an agent-based BPM system we can sum up the major advantages of such an approach as follows [4,15]:

- Agents allow decentralized ownership of tasks, information and resources involved in business processes;
- The use of communicating agents, which also concerns about business logic, allows flexible and dynamic solution paths to the business process execution;
- The adoption of agents provides a high degree of natural concurrency when many interrelated tasks are running at any given point of the business process;
- The decoupling of the parts of the system that agents ensure allows them to be swapped out, replaced, or even added to the system without impacting other parts; and
- Agent technologies are today ready to build highly decentralized and distributed systems with notable non-functional features in terms of solidity and robustness.

Unfortunately the literature has already identified some disadvantages of the promising agent-based approach to the realization of BPM systems (see, e.g., [15]). We summarize the most prominent here for the sake of completeness:

- Agent-based systems have no overall system controller, which implies that the agent-based approach might not be the best choice for managing business processes with a lot of global constraints to be satisfied; and
- Agent-based systems have no global complete knowledge, i.e., an agent's actions are 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. All in all, such issues and their importance originate from the common understanding that considers agent-based systems useful only in a limited set of contexts that are characterized by intrinsic dynamism and uncertainty. This is no longer the case as the agent technologies of today have already proved their maturity and their valuable role in the design and realization of solid and robust software.

3 WADE as a BPM Platform

Obviously not all operative environments are so critical to make apparent the shortcomings of agent technologies and often the benefits that agent-based BPM brings are more relevant than the related issues. In fact, our experiences in real-world BPM suggest that agent technologies can work effectively in traditional settings and that they can provide notable benefits to common tasks of BPM. The work presented in this paper is precisely motivated by such a point of view: we think that agent technologies are now ready to deliver very solid, scalable and visually programmable software systems even in traditional environments where dynamism and uncertainty are not major issues.

3.1 Aims and Scope of WADE

We have been using agent technologies frequently in traditional operative environments and users appreciated such a choice for its maturity and effectiveness in the provision of non-functional features coupled with the possibility of visually programming complex behaviors. The heart of all our experience is *WADE* (*Workflow and Agent Development Environment*) [7], a software tool for the visual development of agents as workflow executors that since version 3.0 can be considered a first class agent-based BPM system. Actually, WADE has already been successfully adopted in a number of mission-critical software systems, as detailed further at the end of this paper, for the possibilities it provides in the visual realization of solutions with distinguished non-functional requirements in terms of scalability and robustness. The role of WADE agents in such systems is not only about exploiting the autonomy of agents in the management of dynamic and unforeseen situations; rather it is about providing developers with friendly tools that represent a robust shield against the complexity of non-functional requirements. Moreover, the tight integration of WADE with mainstream development technologies, like Java and Web services, allows developers incrementally adopting agent technologies in their systems. The parts of the system that can fruitfully empower the features of agents are easily developed using WADE, while other parts are still developed using mainstream technologies with no effort needed for integration.

More in details, WADE is an open-source framework meant to develop distributed and decentralized applications based on the agent paradigm and the workflow metaphor. It is built on top of JADE [3,14], the popular open-source middleware for the implementations of multi-agents systems in compliance with FIPA specifications [9]. WADE adds to JADE the possibility to define agent tasks in terms of workflows and a set of mechanisms to handle the complexity of administration and fault tolerance operations in a decentralized and distributed environment. In the view of a system that WADE advocates, each agent is equipped with a set of workflows and the principal duty of an agent is to enact the proper workflow depending on the dynamic situations it faces. Such workflows are normally described using the pleasant visual notation of the *WOLF* (*WOrkflow LiFe cycle management environment*) graphical editor (see also [1,2] for similar graphical languages for agent-based workflows). WOLF promotes a high-level view of workflows, i.e., of the tasks of agents, and it gives developers a friendly tool shown in Figure 1.

It is worth noting that the conception of WADE and related tools is always concerned about the smooth transition from mainstream technologies and agent technologies in order to ensure a proper management of all well/known issues related to a substantial paradigm shift. This is the reason why WOLF tightly couples the graphical view of a workflow with the underlying Java class that concretely implements it. The developer is free to work on either the graphical view or the Java class freely and he or she can change its work approach as easily as clicking on a tab because WOLF ensures a real-time roundtrip accordance of the two views. Moreover, the choice of implementing WOLF as Eclipse plug-in [8]

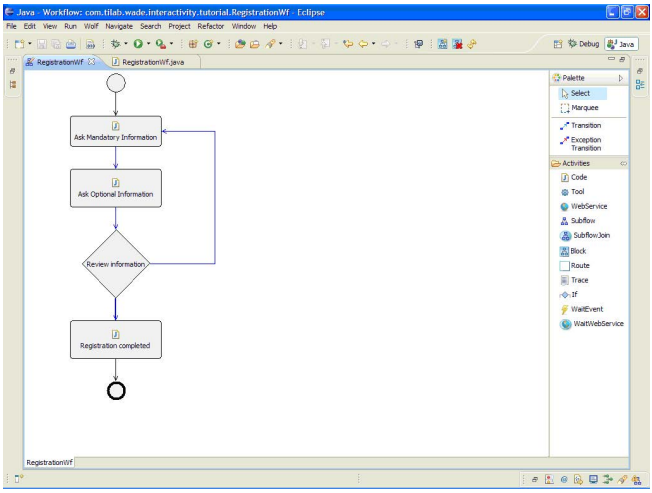

Fig. 1. WOLF visual editor for WADE workflows

that fully exploits the features of the Eclipse platform ensures that developers are allowed to work in one of the most appreciated environments available today.

Many advantages have been demonstrated to become effective once we decide to follow the WADE approach and, among them, it is worth mentioning the possibility of having a graphical representation of a workflow which is 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 can even contribute to the actual development of the system with no need of programming skills.

3.2 BPM-Oriented Evolutions of WADE

WADE was initially conceived back in 2006 to exploit the workflow approach in the implementations of 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 they can be defined in terms of the activities to be executed, the relations between such activities, which specify the execution flow, and the conditions of start-up and termination. 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, e.g., XPDL, BPEL, WS-BPEL [19], 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 internal logic of a given software system. A general-purpose programming language like Java is definitely more powerful and flexible to deal with data transformations, computations and with the low level auxiliary operations that are often needed when specifying the internal logic of the system under development. Then, given that workflows start and terminate their executions in a short time, no persistency mechanism was considered necessary and workflows did not survive to the shutdown of their WADE platform.

Starting from 2010 new requirements from Telecom Italia WADE-based systems, as well as from the open-source community, showed that, though very effective for a particular type of applications, the advocated approach restricted too tightly the actual uses 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 a substantial role in agent-based BPM contexts.

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. In details, if the platform is shut down just after 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 state 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 developed and 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.

Asynchronous Events. Another major step 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. In details, when the process enters such a synchronization activity, the related agent thread is released to prevent resource consumption and the WSMA switches the workflow state from ACTIVE to SUSPENDED. A dedicated API is then provided to allow agents submitting events to the event system. As soon as an event matching the template specified in the synchronization activity is submitted, the workflow agent is resumed and the state of the workflow is switched back to ACTIVE. Furthermore, the information that the event bares 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 it immediately steps forward. It should be noted that the possibility of blocking to receive asynchronous events is not strictly related to long-running workflows however, if the system is restarted, long-running workflows will be recovered transparently and all suspended short-running workflows are immediately aborted.

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 dual possibility of exposing Web services. Such a new feature is twofold. First, 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 exposure feature with the support for asynchronous event described previously. An ad-hoc *WaitWebService* synchronization activity is now available that, when reached, blocks the workflow 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 that is submitted to the event system. Second, it is now possible to automatically expose a workflow as a Web service. The workflow name is mapped to the service name and a single execute operation is generated with input parameters matching workflow's ones. The code intended to serve the invocation triggers the execution of the workflow.

From the architectural point of view, the Web service exposure feature is implemented by a new component called *WadeServices*. This is a common Web application that can be executed within any servlet container, e.g., Apache Tomcat.

It is worth noting that this new feature of WADE is made available in WOLF by means of a simple point-and-click on the workflow Java class.

Administration Web Console. According to the new evolutions of WADE and in order to facilitate the administration of the platform, a Web monitor and configuration console was developed to allow performing both low level management operations, e.g., the start-up/shut-down of the platform, and high-level actions related to the business logics, e.g., browsing and launching a workflow. This new Web console was implemented using the ZK framework [23], an open source solution to develop Web applications based on AJAX machinery. In particular, the ZK framework has been extended to support new ZK components specifically intended to support WADE administration functionalities. Such components, developed for and used by the Web console, can be also reused inside custom Web applications that need to integrate WADE platform management functionality.

4 WADE User-Centric Workflows

The extensive use of WADE in mission-critical applications (see the concluding section and [7] for some examples) has witnessed the notable importance of user interactions in the scope of workflows. This is not surprising and we acknowledge that the idea of workflows has its origins in the management of the work of people. Nonetheless, we believe that the common approach of treating user interactions as yet another type of event does not adequately capture the importance and the high frequency of them. So called user-centric workflows are therefore introduced in WADE version 3.0 as a means to capture workflows that *(i)* frequently need to interact with users, and *(ii)* are mainly intended to gather information and provide feedback to users. WADE now lifts user interactions to a higher level and it provides specific tools and features to manage them effectively. The design guidelines for such a recent development of WADE are as follows:

- The description of the information to provide to users and the related input to acquire from users must be independent of the device that the user is concretely accessing;
- Any element of such a description must be extensible in order to let developers provide more specific descriptions of both input and output information;
- The software application that the user accesses must be replaceable by any custom application once the communication with the WADE platform is correctly performed; and
- No device is privileged and developers must be able to describe workflows in full generality, if they really want.

From such very generic guidelines we choose the *Model-View-Controller* (*MVC*) [10] architectural design pattern as the coarse grained model around which we designed the new interactivity package of WADE. Therefore, WADE version 3.0 adopts the following terminology:

- The model of the interaction is the abstract description of the information to provide to users and the related input expected from users; and
- The view of the interaction is the visual representation of the model realized by the application that the user adopts to connect to a workflow. In concrete terms, the new interactivity package of WADE provides the Java classes of the model and a number of visualizers intended to be integrated in the application shipped to users.

In order to fully exploit the power of user-centric business processes, the developer of a workflow should first inform WADE that the workflow itself needs to interact with users. This is accomplished by realizing a workflow class that extends the `InteractiveWorkflow` class rather than the common `Workflow` class. Such an `InteractiveWorkflow` class is a specific subclass of `Workflow` that provides the needed machinery to link a workflow instance to a visualizer. WADE ensures a one-to-one correspondence between a user and an instance of

an `InteractiveWorkflow` and therefore an `InteractiveWorkflow` has just one user at a time. Multiple users can be accommodated into a single instance of a workflow by having the control of the workflow passed between users, just like we normally do in user-centric workflows.

When an `InteractiveWorkflow` is connected to a visualizer, it is requested to provide the visualizer with a description of the information to present to the user and with a related description of the possible user inputs. Such a mechanism is concretely driven by the workflow developer who can freely use the new method `interact()` that `InteractiveWorkflow` provides. Such a method is supplied with an `Interaction` object that contains the following parts:

- An abstract description of the information to be presented to the user with some abstract requirements on the way information is presented, e.g., by indicating how a set of labels should be aligned on the user screen;
- An abstract description of the information that the user is allowed to return in his or her response;
- An abstract description of the constraints that the user response must meet to be considered valid; and
- A list of possible abstract actions that the user is allowed to choose as valid responses.

Upon executing the `interact()` method, the workflows is put into a SUSPENDED state to allow the corresponding visualizer to present the information to the user and to enable the user to provide feedback by means of one of the available response actions. The visualizer is on duty for showing the information in the best possible way and for allowing the user to provide its response. The visualizer is also responsible for the correctness of the provided response because it is in charge of checking the constraints that identify valid responses. Once the user has validly compiled its response and chosen one of the available response actions, the visualizer returns user response to the workflow instance in terms of a copy of the original `Interaction` object that now contains relevant user input and from which the developer can extract the user response easily. Such an approach allows developers retrieving response information from where they originally decided they should be contained. Moreover, it ensures no redundant information is sent back in responses.

The interaction between the workflow and the visualizer relies completely on WADE agents because *(i)* WADE ensures that any running workflow instance is associated with an agent; and *(ii)* visualizers are implemented by means of WADE agents that can interact with user devices.

How WADE agents are concretely connected with user devices is heavily dependent on the actual visualizer the user is accessing, as discussed briefly later in this section.

4.1 A Model of Interactions

In the WADE nomenclature an interaction is both an abstract description of the information to be provided to users and a means to allow users constructing

responses. Therefore, WADE provides a set of Java classes that are used to describe interactions with such a dual meaning. Such classes are designed using the standard approach adopted in modern user interfaces and they are structured in a containment tree. Figure 2 sketches a class diagram of some of the classes that the developer can use to create a model. Such classes are divided into the following major groups:

- Passive elements, e.g., labels and pictures, that are leafs of the containment tree intended to describe the information to be provided to users;
- Information elements, e.g., text areas and menus of various types, that are leafs of the containment tree and that are meant to provide the user with a means to compile his or her responses;
- Containers, e.g., list and grid panels, that are designed to aggregate a group of children in order to describe their relative position in an abstract manner;
- Actions that describe the types of responses the user can select; and
- Constraints that concretely provide check procedures to ensure the correctness of the user responses.

With the notable exception of constraints, all such Java classes are purely descriptive and they are simple containers for information flowing between an InteractiveWorkflow and a visualizer. They are designed to maintain the clear separation of concerns of the MVC design pattern. All such classes describe the model of an interaction, while the relative controller is implemented by the adopted visualizer, which also generates on the fly the relative view. Such an approach ensures, among other things, that developers are free to add new visualizers and that no visualizer is privileged.

Constraints are peculiar in the scope of the MVC pattern because they are intended to validate user input. They represent a pluggable part of the controller because they are responsible for updating the view upon changes in the model, e.g., by marking invalid components with an error notification. WADE provides a set of general purpose constraints that can be used, e.g., to make sure a mandatory menu has at least an item selected or to warrantee that the text in a text field conforms to a given regular expression.

Finally, it is worth mentioning that WADE version 3.1 already provides descriptors for visual elements that are not supposed to be available to all visualizers. This is the case, e.g., of the `Position` and of the `Camera` classes that are now available only to the Android visualizer. This is not contradictory with the abstract and extensible approach that we enforced with the mentioned design guidelines. In fact, no visual element is guaranteed to be available to all visualizers even if the most common of them are likely to be always there. It is up to the developers of the application that integrates WADE workflows to ensure that workflows use components that are actually available.

4.2 Available Visualizers

At the time of writing WADE provides three visualizers meant to accommodate two important classes of users: Web users and Android users. Web users

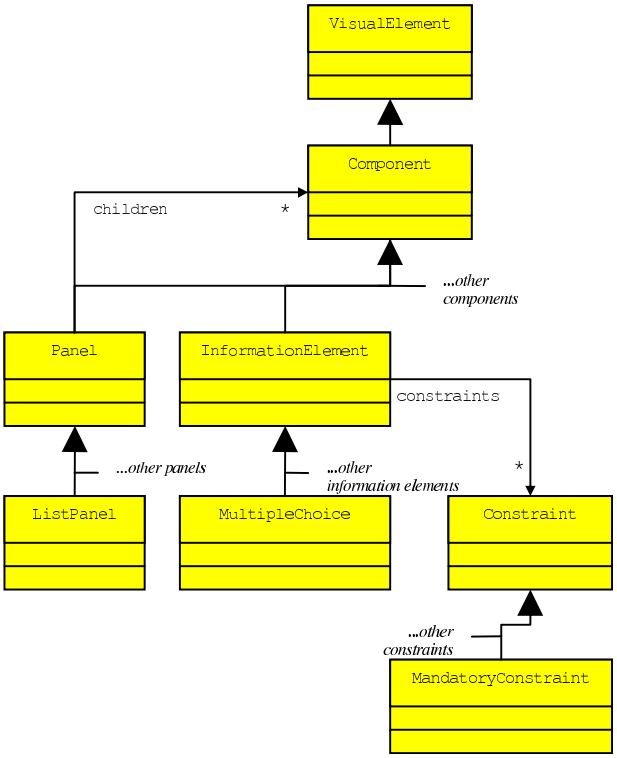

Fig. 2. Excerpt from WADE interactivity package

are allowed to activate new interactive workflows and to connect to suspended workflows by means of dedicated visualizers developed using the ZK toolkit [23] and the *GWT* (*Google Web Toolkit*) [11].

ZK is a very popular toolkit to develop AJAX applications in Java and it is easily interfaced with WADE[1]. The ZK visualizer instantiates one JADE agent on the server side of the Web application for each and every Web session and it ensures agents are properly connected with the WADE platform. The client side of the ZK application is meant to *(i)* present information to the user, *(ii)* provide selectable actions in terms of buttons, and *(iii)* ensure constraints are met before passing any response to workflow agents. The chosen approach ensures a lightweight client that is only in charge of realizing the user interface on the fly and of validating constraints. ZK provides a proprietary communication means between the client browser and the server side of the application that is completely hidden in the deep internals of ZK, thus becoming transparent to developers.

[1] The WADE administration Web console is also developed with ZK.

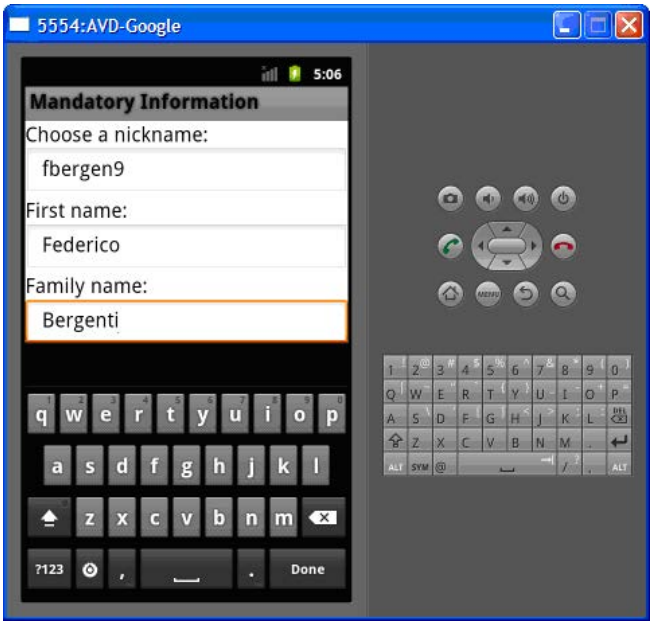

Fig. 3. Android emulator presenting the user interface of a WADE interactive workflow

GWT is the Google's proposal to develop highly interactive Web applications and it provides almost unique means to develop client-side Java code. The GWT visualizer is essentially a porting of the ZK visualizer, even if a clearer separation between the client code and server code can be accomplished. Just like any other GWT application, the visualizer is split into a client side that is responsible to provide a rich GUI to the user, and a server side that communicates with back-end agents and progresses workflows. The two side are linked via GWT RCP mechanism and JADE agents are confined on the server side. Therefore, the actual logic of the visualizer resides on the server side, while the client side is only responsible for building user interfaces and for collecting user input which is then passed back to workflows.

The Android visualizer is developed along the lines of the two Web visualizers and we ensured that the internals of the two visualizers are developed using the same architecture and adopting closely related classes. The major difference with Web visualizers is that the Android visualizer is a single Android application that hosts on the user terminal:

- A JADE container in split mode (see JADE documentation for details [14]) which is created in the scope of the WADE platform;
- The agent needed to connect the user with the workflow; and
- The visual components that are used to dynamically assemble and render the user interface.

No proprietary communication mechanism is needed in this case because the agent and the visual components share some memory of the terminal.

Figure 3 shows the Android emulator presenting the user interface of the sample interactive workflow depicted in Figure 1.

5 Conclusions

This paper presents recent developments of WADE in the larger scope of agent-based BPM. It gives an overview of the main concepts of agent-based BPM and it emphasizes the main features of it in comparison with traditional BPM. Even if we acknowledge the power and possibilities of agent-based BPM to tackle dynamic and unforeseen situations, we advocate the use of agent technologies in the development of BPM systems for their solid non-functional features and for their proven ease of use. Such features are easily understood and mastered by the personnel involved in BPM projects and they ensure a smooth transition from mainstream approaches to innovative technologies with minimal, or even null, issues. Actually, WADE has been appreciated in the development of mission critical agent-based BPM systems for the agile approach that it brings in. WADE provides 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 cores of real systems, thus reducing time-to-market and improving the overall qualities of systems and of development processes.

WADE is commonly used in Telecom Italia for a number of mission critical systems [7,21] that are now in everyday use with real users and in the scope of projects with real customers. The following is a brief list of the most notable initiatives that use WADE in Telecom Italia:

- 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 mash-up platform for service-oriented architectures, enables 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 *SAAS (Software As A Service)* offer for Utilities customers in the fields of electricity, gas and water. This offer includes various systems based on the new functionalities of WADE 3.1 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.: Building a Multi-Agent System from a User Workflow Specification. In: Proc. Workshop "Dagli Oggetti agli Agenti". CEUR Workshop Proceedings, vol. 204 (2006)
2. Bartocci, E., Corradini, F., Merelli, E., Scortichini, L.: BioWMS: A Web-based Workflow Management System for Bioinformatics. BMC Bioinformatics 8(S-1) (2007)
3. Bellifemine, F.L., Caire, G., Greenwood, D.: Developing Multi-Agent Systems with JADE. John Wiley & Sons (2007)
4. Bolcer, G.A., Taylor, R.N.: Advanced Workflow Management Technologies. Software Process: Improvement and Practice 4(3), 125–171 (1998)
5. BPMN – Business Process Modeling Notation, `http://www.bpmn.org`
6. Cai, T., Gloor, P.A., Nog, S.: DartFlow: A Workflow Management System on the Web Using Transportable Agents. Technical Report, Dartmouth College (1997)
7. Caire, G., Gotta, D., Banzi, M.: WADE: A Software Platform to Develop Mission Critical Applications Exploiting Agents and Workflows. In: Proc. 7th Int'l Conf. Autonomous Agents and Multiagent Systems, pp. 29–36 (2008)
8. Eclipse, `http://www.eclipse.org`
9. FIPA – Foundation for Intelligent Physical Agents, `http://www.fipa.org`
10. Fowler, M.: Patterns of Enterprise Application Architecture. Addison-Wesley (2003)
11. Google Web Toolkit, `http://code.google.com`
12. Grundspenkis, J., Pozdnyakov, D.: An Overview of the Agent.Based Systems for the Business Process Management. In: Proc. Int'l Conf. Computer Systems and Technologies (2006)
13. Hawryszkiewycz, I., Debenham, J.: A Workflow System Based on Agents. In: Quirchmayr, G., Bench-Capon, T.J.M., Schweighofer, E. (eds.) DEXA 1998. LNCS, vol. 1460, pp. 135–144. Springer, Heidelberg (1998)
14. JADE – Java Agent Development framework, `http://jade.tilab.com`
15. 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(2-3), 105–130 (1996)
16. Jin, W., Chang, S.T.: Agent-based Workflow: TRP Support Environment (TSE). Computer Networks and ISDN Systems 28(7-11), 1501–1511 (1996)
17. Pang, G.: Implementation of an Agent-Based Business Process. Technical Report, University of Zurich (2000)
18. Pesic, M., Schonenberg, H., Van der Aalst, W.M.P.: DECLARE: Full Support for Loosely-Structured Processes. In: Proc. 11th IEEE International Enterprise Distributed Object Computing Conference, p. 287 (2007)
19. Shapiro, R: A Comparison of XPDL, BPML and BPEL4WS (Rough Draft), Cape Vision (2002)
20. Telang, P.R., Singh, M.P.: Specifying and Verifying Cross-Organizational Business Models: An agent-Oriented Approach. IEEE Transactions on Services Computing 5(3), 305–318 (2012)
21. Trione, L., Long, D., Gotta, D., Sacchi, G.: Wizard, WeMash, WADE: Unleash the Power of Collective Intelligence. In: Proc. 8th Int'l Conf. Autonomous Agents and Multiagent Systems (2009)
22. Workflow Management Coalition. Workflow Management Coalition Terminology & Glossary, `http://www.wfmc.org`
23. ZK Open-Source Framework, `http://www.zkoss.org`