An Implementation Model for Agile Business Process Tools

António Rito Silva[1], David Martinho[1],
Ademar Aguiar[2], Nuno Flores[2], Filipe Correia[2] and Hugo Sereno Ferreira[2]

[1] IST/UTL
Center for Organizational Design and Engineering - INOV
Rua Alves Redol 9
Lisbon, Portugal
{rito.silva,david.martinho}@inov.pt
[2] INESC Porto
Faculdade de Engenharia da Universidade do Porto
Rua Dr. Roberto Frias
Porto, Portugal
{ademar.aguiar,nuno.flores,filipe.correia,hugosf}@fe.up.pt

Abstract. In todays changing environments, organizational design must take into account the fact that business processes are incomplete by nature and that they should be managed in such a way that they do not restrain human intervention. Current approaches to Business Process Management (BPM) intend to perceive the business as it is and to model how it should be in a single step, completely describing the business processes. In this paper we propose an implementation model for an incremental and bottom-up approach to BPM grounded in business process tools that integrate the modeling and execution of incomplete business processes. To support collaboration, the tools embed social software features, such as wiki-like features, in the modeling and execution tools of business processes. The implementation model supports business process modeling at the instance level, promoting of instance cases to types and a wiki-like interface to business process modeling.

Key words: Business Process Management, Agile, Modeling and Execution Environment, Social Software, Wiki

1 Introduction

In today's business environment, characterized by non-stop and fast occurring change, it is very hard to follow an AS-IS/TO-BE approach to Business Process Management. AS-IS/TO-BE approaches assume a strategy which requires business processes to be completely described, both AS-IS and TO-BE, before any intervention can begin (either technological or managerial). This gives rise to lengthy modeling activities aiming at capturing a complete model of both the existing business processes (AS-IS) and the new business processes (TO-BE)[Weske, 2007].

There are several reasons why AS-IS/TO-BE approaches do not work as well as they should:

- Different people have different perspectives on processes: top managers have a high-level perspective while users have more detailed perspectives. IT consultants, on the other hand, have a systems-slanted view of the same processes. As a result, it is difficult to get all the players to agree on what the process definitions are.
- Process definition is driven by the organization's institutional strategies, policies and procedures and does not take into account the tacit knowledge users deploy in operating the real organization. Tacit knowledge is mostly gathered on a case-by-case approach instead of on institutional rules and procedures.
- The organization and its rules and structures are constantly emerging and changing. This requires intervals of (very) short duration between process modeling and implementation.
- TO-BE approaches follow mechanistic models of planned change that, as a whole, restrict collaboration and reduce the empowerment of people, disengaging them from organizational responsibilities and delegating intelligence to the Information Systems (IS).
- Business processes contain many variations that take time to identify and increase dramatically the complexity of the model. Furthermore, most of these variations are exceptions that only occur in a few business process instances.[Russell *et al.* , 2006; Golani *et al.* , 2007]
- A technology-free business process modeling is a naïve approach because it ignores that there is an entangling between coordination of people and the technology they use in the execution of the business processes. Modeling business processes, in the absence of a technological perspective, results in processes that do not fit with organizations real praxis.

To overcome the mentioned limitations of AS-IS/TO-BE we propose that new approaches should be characterized by short feedback cycles [Beck *et al.* , 2001]. In their proposals about Organizational Design and Engineering (ODE), Magalhaes & Rito-Silva [2009] suggest that organizational development projects (i.e. projects involving design and engineering activities) should be planned and executed through a series of small activities of short duration, such that after each intervention a new observation is carried out to identify how the organization was changed by the last intervention. The organizational routines contained in the computer-based artifacts provide the required stability for observation points to be created. Instead of strategic alignment of IS/IT, those authors propose organizational steering. Steering emphasizes continuous analyses through observation of the organization's evolution, making small adjustments between interventions, in moving the organization towards the goals defined by the strategy. The engineering activities should be of short duration followed by the artifacts integration in the organization, where design is a continuous activity and not just a starting point but also an ever-changing destination.

Following the ODE tenets and considering the problems of AS-IS/TO-BE approaches, an agile business process proposal should be characterized by:

Incompleteness

- Business processes do not need to be completely understood. Trying to completely understand a process is time expensive, reducing the number of feedback cycles and increasing the chances that once implemented, the process no longer conforms to the organizational needs.
- The process does not need to be completely specified in the sense that it is allowed that activities not pre-specified occur in some of its instances. This allows the instantaneous adaptation of executing processes to the emergence of new organizational needs.
- Incompletely defined processes should be executable and their execution integrates specified with non-specified activities such that incompletely specified processes do not limit the normal operation of the organization.

Empower people

- Processes should promote collaboration, creativity and intelligence instead of restricting them.
- The system should allow people to execute non-specified activities and combine their execution with specified activities.

Business process definition integrated with technologic use

- To avoid a situation of paralysis by analysis due to different perspectives on processes, process definition should be integrated with the operation of the business. This way, the different perspectives on process definition will leverage on the actual operation of the business.
- Integrate the execution and definition of the process, such that the process executor is also one of its modelers, thus avoiding the technology-free definition of processes.

Definition on a case-by-case approach

- It should be possible to describe processes on a case-by-case approach instead of trying to specify all the possible situations in a single specification. This will allow a reduction of the complexity of business process definitions.
- It should be possible to promote unexpected exceptions, described on a case-by-case basis, to become part of the specification of the business process. This approach promotes the bottom-up definition of business processes.

This article proposes an implementation model, following a new AGILe busIness PrOcess management (AGILIPO) approach, which fulfills the identified requirements. The next section describes the AGILIPO approach.

2 Agile Business Process Management

Considering the set of requirements identified above, we propose an agile business process approach for bottom-up modeling and implementation of incomplete

business processes. AGILIPO follows the principles of agile software development [Beck *et al.* , 2001] and of organizational design and engineering [Magalhaes & Rito-Silva, 2009]. AGILIPO is supported by collaborative modeling and execution tools that embed social software-like functionalities. The distinctive feature of AGILIPO tools is the integration of modeling with execution activities blurring the differences between definition and operation of business processes. While executing a particular instance of an incomplete business process, users are empowered to execute, on a case-by-case basis, activities that are not specified.

An incomplete process definition is specified by a set of activities that describe part, but not all, of the process instances behavior. A process instance contains activity instances which can be either specified or non-specified. An activity definition can either be automated, when it includes the interaction with external applications, even requiring the user participation, or non-automated. Automated activities contain hardcoded functionality and require programming activities to implement them.

AGILIPO considers three different roles that business process management stakeholders can take: executor, modeler and developer. The executor is able to conduct business process execution either by making use of specified activities or create non-specified activity instances whenever the specified ones cannot fulfill the current execution situation. The modeler is capable of changing the business process model, specifying new non-automated activities. The developer may rely on these non-automated activities and automate them by coding the interaction with external systems. Note that executors, modelers and developers are roles that can be played by the same person.

When executors create non-specified activity instances they are contributing to the business process model following a case-by-case approach. The non-specified activity instances capture business process variations and exceptions, allowing the process instance adaptation without requiring all possible situations to be specified in the process model. Moreover, process instance adaptation occurs in the context of process execution, where non-specified activity instances are integrated with instances of specified activities. Afterwards, executors can tag non-specified activities and participate in the creation of ontologies for the business process. Following a folksonomy approach, executors tag activities, share their tags, and search for activities based on these tags.[Marlow *et al.* , 2006; Wal, 2007; Wu *et al.* , 2006]

Modelers analyze the set of non-specified activity instances with its associated folksonomy, and generalize the variations over the existing business process model, synthesizing a new version of the model. Once integrated in the model, modeler's suggestions enrich the set of specified activities although in a non-automated form. Afterwards, developers rely on such suggestions to automate the non-automated activities. Developers' decision on which non-automated activities to implement is driven by cost-effectiveness. Cost-effectiveness is determined by the frequency of activity occurrence in process instances and by the time consumption in activity execution.

Model evolution is a concern of AGILIPO tools. Suggestions are synthesized by modelers following a wiki-like approach[Cunningham, 2002], where new suggestions leverage on previous ones, thus creating new revisions of the model. In this way, AGILIPO intends to foster a knowledge creation process, organically and incrementally (Wikipedia-like)[Riehle, 2008], where contributors are motivated to participate in the modeling of an incomplete process by reading contributions of others and continuously adding their own knowledge[Garud *et al.* , 2008].

As an example, consider an online book-selling store with the process *MakeSell* having three known and specified activities: *AddBookToOrder* followed by either *PayWithCheck* or *PayWithCreditCard*. However, as a client goes directly to the physical store and wishes to pay with cash, there is no activity that will cover such situation. The employee may then create an instance of a non-specified activity and associate it to the current instance of the *MakeSell* process. Afterwards, the employee needs to assign the activity instance to a supervisor because she does not have enough authority to receive the money herself. Therefore, the employee, instead of executing the activity instance, addresses it to its supervisor. The supervisor is then able to execute the activity instance created by the employee and finish the *MakeSell* process. The specified *MakeSell* process instance is terminated having two different types of activity instances: an instance of the specified *AddBookToOrder* activity, and an instance of a non-specified activity to cover paying with cash exceptional situation. As can be seen, AGILIPO empowers people to perform business processes according to their tacit knowledge and allows responsibility delegation based on the roles played by the organization members: it is up to the employee to know that in this exceptional situation only the supervisor can receive the money.

Both, employee and supervisor can tag the non-specified activity instance with keywords like for example *Pay*, *Books*, *Money* and *Cash*. That way, executors that in the future get caught in the middle of such exceptional case, could easily find similar occurrences while searching by those tags and easily make use of the same activity instance structure. Moreover, if further executions of the *PayWithCash* activity take place in the same context, and have similar tags, then a modeler can specify the *PayWithCash* activity in the model as a non-automated activity. This would create a new variation, explicitly represented in the model, such that the employee does not need to search for similar exceptions: the exceptional behavior becomes a suggestion. Finally, the non-automated suggested activity *PayWithCash* could be hardcoded into the application model by a developer automating for example the delegation procedure.

In synthesis, AGILIPO tools support the modeling and execution of business processes, using a wiki-like interface to read and update the processes, integrates automated and non-automated parts of the process, supports the execution and modeling of exceptional behavior, and enforces a continuous knowledge creation process around incompletely defined and understood processes.

3 AGILIPO Model

The AGILE BPM approach is rooted in AGILIPO tools, which provide an integrated environment for business processes modeling and execution. The integrated environment embeds social software features for business process tagging, folksonomy construction and suggestions integration. These tools are built on top of a model, the AGILIPO Model. The AGILIPO Model is based on a business process model supporting incomplete processes. These processes are underspecified and integrate the specified part of the model with modelers' suggestions on business process evolution. A distinctive feature of the AGILIPO model is the smooth integration of non-specified activity instances, automated activity definitions and non-automated activity definitions such that users having different roles can collaborate in the modeling and execution of business processes.

3.1 Business Process Execution Model

At the kernel of the AGILIPO Tools is an execution environment for incomplete business processes. Underspecified business processes should allow the flexible incremental definition of new activities. The Case Handling [Aalst *et al.* , 2005] paradigm for business process execution is proposed as the basis of AGILIPO execution model, because it decouples the activities flow of control from the activities themselves, simplifying addition and removal of activities.

The Case Handling concepts are:

- **Case (Process)** - represents a process composed by a collection of activities. The process instances include an ordered log of the activities already executed in the instance.
- **Activity** - represents an atomic action that is enabled for execution if its conditions are verified. It can also have data objects associated, which may be used in the conditions. An activity execution may include the interaction with an external service, as a data base service.
- **Condition** - is a boolean statement that enables an activity to be executed when its value is true. Conditions may use process data objects or domain specific information such as, for instance, the information in an application of human resources.
- **Data Object** - is a data field the user must fill in when executing an activity. Data Objects are usually implemented as data types such as, for instance, number, money, string, location, file, photo, date, and so on. Data objects definitions, together with conditions determine the flow of control.

In Figure 1 we can see an illustration of the Business Process Execution Model using a Case-Handling approach.

The case-handling concepts are implemented as abstract classes, and specific business processes are implemented by defining subclasses of the abstract classes. We can see this execution model in Figure 2. All the instances of a given specific subclass have the same definition. Specific subclasses contain domain logic and can trigger invocations on external systems. They are hardcoded on the model

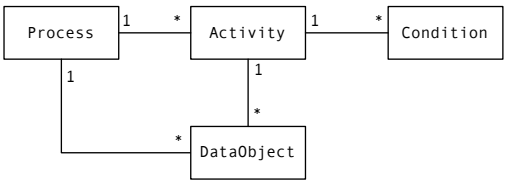

Fig. 1. Business Process Execution Model using Case-Handling approach

as implementations that might have specific behavior (e.g. running queries on databases or calling webservices). So, specific subclasses belong to the automated part of the business process model. For instance, if the process has an activity *BuyPaper*, executing such activity may imply calling a webservice on the paper supplier to place an order.

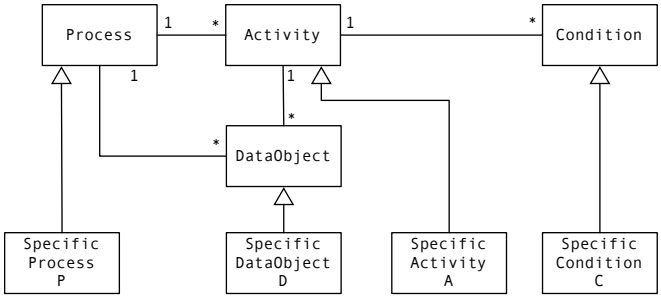

Fig. 2. Execution Model

3.2 Execution of Incomplete Business Processes

The execution of incomplete processes should not restrain the normal operation of the business. This means that users can execute non-specified activities, since incomplete processes are underspecified, in the context of the incomplete process instance. To provide the integration of specified and non-specified parts of business processes, the AGILIPO models has the concept of generic class that allows the adaptation of incomplete business process instances. When executing a particular business process instance the user can create an instance of a generic class to proceed with the execution of a non-specified part of the business process.

There is a generic class for each one of the case-handling concepts. As we can see in Figure 3, Generic classes are also implemented as direct subclasses of the case-handling abstract classes but, contrarily to specific subclasses, their instances do not have the same definition: generic classes do not have specific

behavior hardcoded. This means that instances of a generic subclass can represent different kinds of entities. For example, an instance of a generic activity can represent a sales activity while another can represent a payment activity.

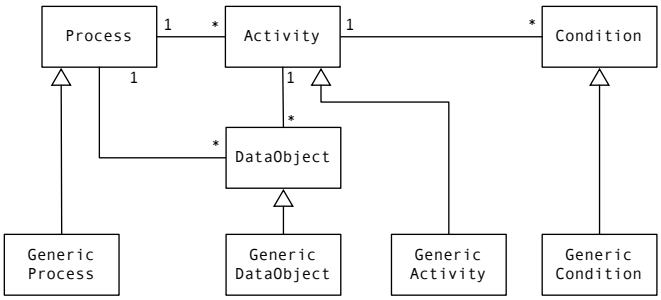

Fig. 3. Generic Entities inheriting from the model

The use of generic classes empowers the user to adapt, during execution, incomplete processes whenever it is needed. However, users should be authenticated and possess the right organizational roles such that it is possible to audit process instances execution. Moreover, identification of the organizational roles is also fundamental to allow the delegation of generic activity instances to a specific user or to a group of users. For instance, an authenticated user should be able to decide that, after finishing the execution of an activity, a next possible activity is a generic one to be performed by a user belonging to the sales department. So, AGILIPO approach requires the identification of each one of the organization members and its set of roles. So, an organization model needs to be defined.

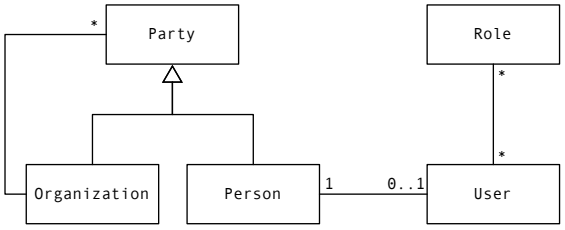

Fig. 4. Organizational Model

The organization model, Figure 4, has an entity called *Party* which is a superclass of two different classes: *Organization* and *Person*. Organizations contain parties, that is, they are composed by sub-organizations and persons. This composition relationship of *Organization* with the abstract *Party* class allows the creation of an organization as a tree-structure where sub-organizations are

intermediate nodes and people are the leaves. The organizational model also considers the concepts of *User* and *Role*, associating a *User* to *Person* and its set of roles. Roles can be explicitly assigned to users or inferred from the organizational structure the user belongs to. The concepts of *User* and *Role* facilitate identification, authentication and authorization.

The proposed organizational model is necessary for the bottom-up definition of business processes. Any organization has a structure, which defines a set of responsibilities, even if it does not have its business processes explicitly defined. The organization's structure is used by business process executors to delegate and assign non-specified activity instances to organization members.

3.3 Non-automated Business Process Types

Instances of generic classes do not have a definition. However, as knowledge about the business process behavior increases, it should be possible to associate a definition to generic class instances: instances of generic classes become part of the business processes specification. To achieve this we apply the *TypeObject* pattern[Johnson & Woolf, 1997], to associate a type with a generic class instance.

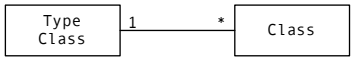

Fig. 5. TypeObject pattern

In Figure 5, there is an illustration of the *TypeObject* pattern which represents the type of an object as an object of type *TypeClass*. This allows the dynamic definition of new types by creating new instances of *TypeClass*.

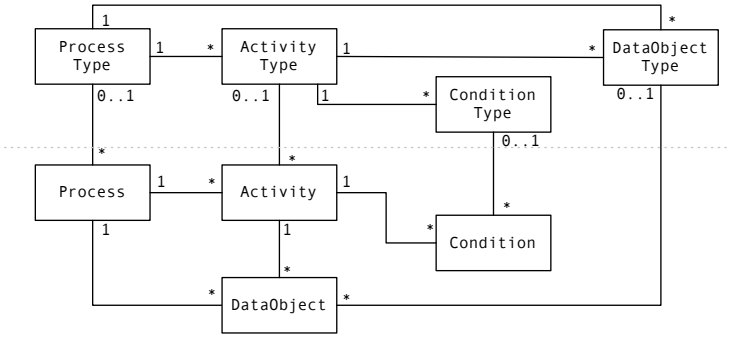

Fig. 6. Type process model

Figure 6 presents the combined model that applies the *TypeObject* pattern to each one the AGILIPO model classes. Note that the *TypeObject* pattern was ex-

tended to allow instances without a type, e.g. instances of *Activity* are associated with zero or one instances of *ActivityType*.

The type process model allows associating types to both specific and generic class instances. Types associated with specific class instances are automated types, where the specific class has hardcoded specific behavior, while types associated with generic class instances are non-automated types, as the generic class does not have any specific behavior.

Figure 7 presents three different possible links between an activity instance and a type object. In case (i) we have an instance of a specific activity linked to the instance that represents its type, *specificActivityA*. This association is automatically created whenever a specific activity is instantiated, thus, all instances of the same specific class should be linked with a unique instance of *ActivityType*, which represents their type. We also pointed that generic activity instances were typeless by default, this is, they have no type associated as all generic activities share the same class *GenericActivity*. This fact is represented by case (ii). Yet, one may create an instance of *ActivityType* to create a new type for an activity. In case (iii), a generic activity instance is said to be of type *activityB*. Note that this new type corresponds to a non-automated type that was suggested and can be related with automated types, as they are all concepts living in the same model. This allows the smooth integration of automated and non-automated types.

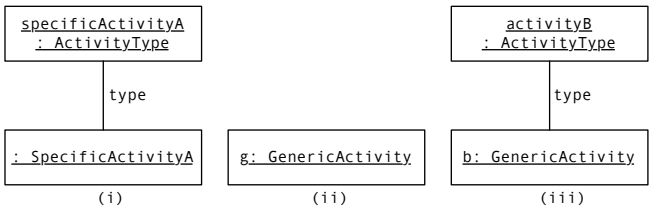

Fig. 7. Object model example

3.4 Business Processes Evolution

Suggestions are the consequence of users experience on the execution of incompletely specified business processes. Thus, to foster the incremental definition of business processes AGILIPO requires a dynamic type model that evolves by integrating suggestions.

The evolution of the type model occurs by creating and deleting instances of type. A new business process is defined by creating a new instance of ProcessType, but to update a business process it is only necessary to create or delete some of its parts, like for instance creating a new instance of ActivityType and associating it with the ProcessType instance.

The evolution of the type model raises the question of what to do with the instances associated with the updated types. These instances do not conform to the new types and need to migrate to the new process definition. However, this may not be always the case. For instance, a business process instance that finished executing according to a previous definition of the business process does not need to migrate to the new definition. So, the type model needs to support several versions of the same business process, and not all the instances need to be associated with the same version of the type model.

TemporalObject pattern [Fowler, 1999] is used on objects that change overtime, considering two roles: one to foster the object continuity and other to support its versioning. In Figure 8 we can see these two roles mapped by two classes. The *Entity* class contains the continuity of the object and has a temporal relationship with the *EntityVersion* class that stores information about the different versions of the object. The temporal stereotype attached to the one-to-one relationship represents that for each instant in time, only one version is valid.

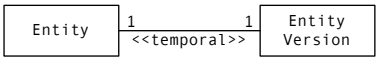

Fig. 8. TemporalObject pattern

The versioning of the type model, illustrated in Figure 9, follows the *TemporalObject* pattern, it associates versions to each of the instances of the type model. Each type instance has a unique identifier that is common to all its versions and each version has a timestamp that establishes the overall order between them. The current version of a type instance is the version with the higher timestamp. When a type instance is created or deleted, a version is also created with the current timestamp. In the case of deletion, the created version is marked as deleted.

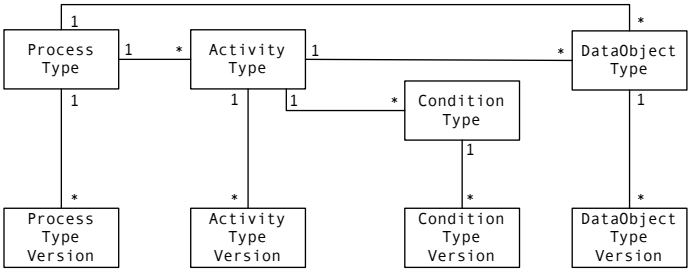

Fig. 9. Versioned Type Model

Consider a business process type instance holding the unique identifiers of each one of its activity type instances and having several versions with different timestamps. The current version of the business process type is given by the most recent versions of its process type instance and of each one of its activity type instances. Nevertheless, when the most recent version of an activity type instance is in state deleted, it will not be part of the current process type definition. Similarly, the instance of a business process type in a given timestamp t is given by the versions with the higher timestamp smaller or equal than t.

Due to type model versioning, none of the previous versions are lost allowing modelers to safely synthesize a unique version of the model by collaboratively evolving from the last version.

Type systems enforce compliance between type definitions and their instances, but in our approach we do not require strict compliance between instances and types since it would forbid business processes evolution. An incomplete business process is, by definition, a process that allows instances to differ from their definition. The level of divergence between a type definition and its instances is a good measure of the need for type evolution.

Both automated and non-automated types can evolve, but automated types require their specific classes to be re-implemented. However, when the specific class is not re-implemented, the execution following the new definition may not be consistent with the automated part of the process. To overcome this problem, the execution should continue to follow the automated type version that complies with the implementation.

3.5 Social-Software Features

AGILIPO strategy for business process modeling is similar to wikipedia's strategy for knowledge gathering [Spek *et al.* , 2006], blurring the distinction between consumers and producers of information. It emphasizes the synthesis of the different suggestions to the business process model through collaborative participation. To foster collaboration among executors, modelers and developers, social-software features are used to promote communication:

- **Wiki-pages** - The AGILIPO model is presented in wiki-like pages enforcing suggestions synthesis.
- **Comments** - Users may express their opinion about a suggested model modification.
- **Ratings** - Being able to quantitatively rate a contribution to quickly analyze overall's opinion instead of synthesizing all the comments.
- **Tagging** - Create folksonomies around the business process instances in order to add semantic value to their content before the business process model evolves.

The presentation and manipulation of the AGILIPO model in terms of wiki-like pages considers type pages and instance pages.

A type page presents the type model for a particular type, e.g. a particular process type. Each page has a version number, contains a description of the types

it is associated with, presents aggregated information about its instances, and has links to its page instances.

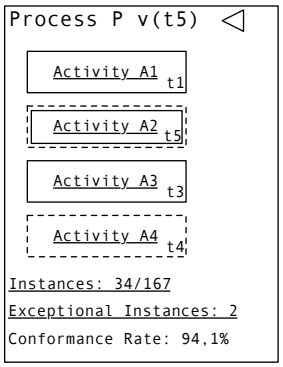

Fig. 10. Process Type Page

Figure 10 shows a *Process P* type page. *Process P* has four specified activity types *A1, A2, A3* and *A4*. *A1* and *A3* are automated types, represented by a solid line, *A4* is a non-automated type, represented by a dashed line, and *A2* is a non-automated type that evolved from a suggestion to a automated type but the implementation was not changed accordingly, which is represented by a inner solid line. Aggregated information about the number of instances that belong to this version of the type and its conformance rate is also presented.

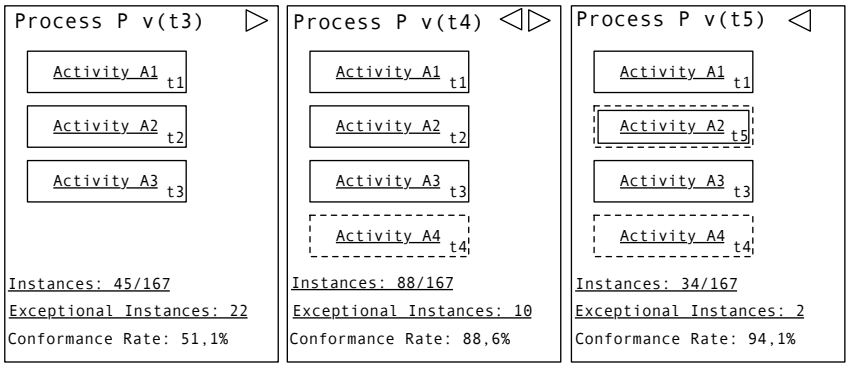

Fig. 11. Process Type Page Evolution

Page types versioning is built on top of AGILIPO model versioning. The Figure 11 presents an example of a page with three versions: the user navigates back and forward through versions by clicking on the arrows.

Activity timestamps are shown to make visible the relation between versioning at the AGILIPO model entities and page versioning. In the first version, *Process P* has three activities *A1, A2* and *A3*, all created at different timestamps and the page version corresponds to timestamp *t3*. The second page version results from adding a new non-automated activity *A4* to the process on timestamp *t4*, the page version corresponds to timestamp *t4*. On the third version, a suggestion on a automated activity was made on a timestamp *t5*, the page version corresponds to timestamp *t5*.

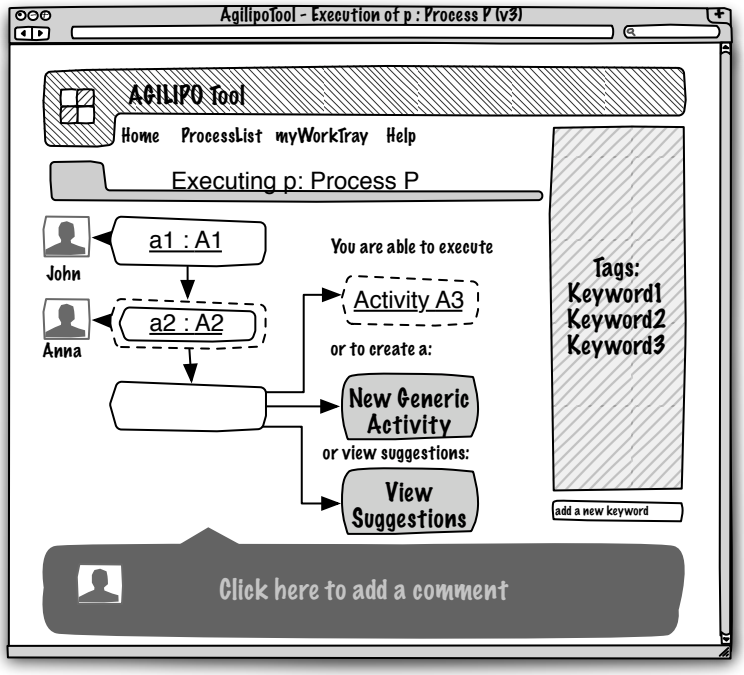

Fig. 12. Process Instance Page

When directly accessed, a type page presents the latest version of the types presented in the page. Contrarily to the usual wiki page navigation, navigation is done in the same timestamp to give a consistent view of the model. For instance, in Figure 11, when at *Process P* page version *t4*, the navigation to activity *A2* type page presents the *A2* activity model, including its conditions and type objects, at timestamp *t4*.

An instance page contains a link to its type page version, in case the instance has a type, contains the log of the instance execution and the next possible executions. Figure 12 represents an instance page of *Process P*. The figure shows the users that executed activities *a1* and *a2* and a list of possible next executions, activity type *A3* and a generic activity. If awareness features are provided, the

page can also suggest the execution of particular activities based on past executions of similar process instances.

Both, type and instance pages, can provide other social-software features, like comments, ratings and tagging. It should be possible to have customized pages aggregating non-structured information provided by users, and structured information from the AGILIPO model. The description of a wiki model that integrates both kinds of information is not in the scope of this article but the use of a Weakly-Typed Wiki [Correia *et al.* , 2009] seems promising.

4 Related and Future Work

Several authors have recognized the need for agile business process management, like Shafii [2008], however, the solutions they propose follow an AGILE Modeling approach [Ambler, 2009] and are strongly committed to Model Driven Engineering (MDE) tools [Schmidt, 2006]. MDE tools, completely or partially, generate the executing system from a model, which requires models to be computationally complete. Contrarily to MDE approaches, which emphasize executable models and automatic code generation, AGILIPO stresses executors, modelers and developers interaction during modeling and execution activities. AGILIPO fosters knowledge gathering without requiring programming skills from all users.

The AGILIPO tools follow some of the design principles of wiki [Cunningham, 2006], focusing on the bottom-up modeling of business process, leveraging on the ideas of ActionBase [ActionBase, 2009] where the execution of exceptional behavior is supported in the context of an email-based tool. Similarly to ActionBase, AGILIPO supports the bottom-up definition of a process model, however, AGILIPO provides an implementation independent business process type model, and, enforces the synthesis of modeler's suggestions.

The execution of business process instances, not complying with a definition, follows the principles of acceptability-oriented computing [Rinard, 2003] that define the region the system must stay within to remain acceptable to the user. To characterize this region for business processes we need to study the existing consistency and conformance criteria between instances and their types like, for instance, the criteria proposed in [Casati, 1999].

To achieve the integration of modeling and execution, together with the support of instance-based modeling and execution, the AGILIPO implementation model is designed to integrate case-handling [Aalst *et al.* , 2005] with Type Object [Johnson & Woolf, 1997] and Temporal Object [Fowler, 1999] patterns. Case-handling addresses the problem that "many processes are much too variable or too complex" [Weske *et al.* , 2004] and the design patterns extend the case-handling model with a versioned adaptive object model [Yoder & Johnson, 2002]. Moreover, the resource perspective of newYAWL [Russell *et al.* , 2008] can be used to support the organizational structure and the delegation of generic activity instances.

AGILIPO fosters knowledge gathering without requiring end-users to possess programming skills. Such knowledge is expressed by the users whenever

they adapt the business processes. To provide expressiveness increase, AGILIPO aims to empower the users with a set of languages to adapt the business processes. Although the generic classes presented earlier may provide such generic mechanism, however, they may be too permissive and lack automation capabilities. This problem may be tackled by the use of Domain Specific Languages (DSL) [Fowler, 2009], which restrict the set of possible adaptation and/or assist users when adapting the business process. Therefore, AGILIPO aims to provide an end-user programming environment to empower the user, and, at the same time, intends to establish a balance between the advantages and disadvantages of using such approach. The definition of such domain specific languages should leverage on the work about configurable and adaptable workflow models [Russell *et al.* , 2006; Schonenberg *et al.* , 2008; Gottschalk *et al.* , 2008]. In particular, it is necessary to evaluate how the worklets approach [Adams *et al.* , 2006] can be used to support business process conditions and how it can be extended to allow conditions adaptation for a particular business process instance.

The AGILIPO implementation model can be integrated with a wiki engine. The Weakly-Typed Wiki [Correia *et al.* , 2009] seems to be a good candidate to integrate the AGILIPO model since it supports incremental formalization of structured pages and type evolution. The current AGILIPO implementation model does not support automation of the business processes at the model level. We intent to experiment the ideas in [Anslow & Riehle, 2008] to evaluate the feasibility of end-user automation of business processes.

5 Conclusions

Agility is being recognized as a crucial new quality for future BPM approaches [Dreiling, 2009]. In this paper we propose a novel approach for agile BPM based on the embedding of social software features into the business process modeling and execution tools. The distinctive feature of these tools promotes process modeling as a continuous activity that is intertwined with process execution activities, fostering a knowledge creation process that blurs the separation between users and designers of business processes.

To accomplish the AGILIPO vision, we are proposing an implementation model that integrates business process features with social software features, and a user interface that preserves the wiki-like usability for business process modeling. The model addresses the set of requirements deemed as necessary for an AGILIPO tool. The proposed model is a foundation that highlights the different concerns that an AGILIPO implementation model has to address. For each of the identified concerns, we point out possible solutions and identify open research issues.

References

Aalst, Wil M.P. van der, Weske, Mathias, & Grünbauer, Dolf. 2005. Case Handling: A New Paradigm for Business Process Support. *Data and Knowledge Engineering*, **53**, 2005.

ActionBase. 2009. *ActionBase*. http://www.actionbase.com/ (accessed June 1, 2009).

Adams, Michael, Ter, Edmond, David, & van der Aalst, Wil. 2006. Worklets: A Service-Oriented Implementation of Dynamic Flexibility in Workflows. *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE*.

Ambler, Scott W. 2009. *AgileModeling*. http://www.agilemodeling.com/ (accessed June 1, 2009).

Anslow, Craig, & Riehle, Dirk. 2008. Towards End-User Programming with Wikis. *In Proceedings of the Fourth Workshop in End-User Software Engineering (WEUSE IV)*.

Beck, Kent, Beedle, Mike, van Bennekum, Arie, Cockburn, Alistair, Cunningham, Ward, Fowler, Martin, Grenning, James, Highsmith, Jim, Hunt, Andrew, Jeffries, Ron, Kern, Jon, Marick, Brian, Martin, Robert C., Mellor, Steve, Schwaber, Ken, Sutherland, Jeff, & Thomas, Dave. 2001. *AgileManifesto*. http://www.agilemanifesto.org (accessed June 1, 2009).

Casati, Fabio. 1999. A discussion on approaches to handling exceptions in workflows. *SIGGROUP Bull*, **20**(3), 3–4.

Correia, Filipe Figueiredo, Ferreira, Hugo Sereno, Flores, Nuno, & Aguiar, Ademar. 2009. Incremental Knowledge Acquisition in Software Development Using a Weakly-Typed Wiki.

Cunningham, W. 2002. *"What is a Wiki". WikiWikiWeb*. http://www.wiki.org/wiki.cgi?WhatIsWiki (accessed June 1, 2009).

Cunningham, W. 2006. Design principles of wiki: how can so little do so much? *WikiSym '06: Proceedings of the 2006 international symposium on Wikis*.

Dreiling, A. 2009. Business Process Management and Semantic Interoperability - Challenges Ahead. *Handbook on Business Process Management - Springer*.

Fowler, Martin. 1999. *Things that change with time*. http://martinfowler.com/ap2/timeNarrative.html (accessed June 1, 2009).

Fowler, Martin. 2009. *Domain Specific Languages*. http://www.martinfowler.com/bliki/DomainSpecificLanguage.html (accessed June 1, 2009).

Garud, Raghu, Jain, Sanjay, & Tuertscher, Philipp. 2008. Incomplete by Design and Designing for Incompleteness. *Organization Studies - SAGE Publications*, **29**(03), 351–371.

Golani, Mati, Gal, Avigdor, & Toch, Eran. 2007. Business Process Management Workshops. *Pages 54–65 of: Business Process Management Workshops*.

Gottschalk, F., van der Aalst, W.M.P., Jansen-Vullers, M.H, & Rosa, M. La. 2008. Configurable Workflow Models. *International Journal of Cooperative Information Systems*, 223–255.

Johnson, Ralph, & Woolf, Bobby. 1997. *Type object: Pattern Languages of Program Design*. Addison-Wesley Longman Publishing Co., Inc.

Magalhaes, R., & Rito-Silva, A. 2009. *Organizational Design and Engineering*. Tech. rept. Center for Organizational Design and Engineering.

Marlow, Cameron, Naaman, Mor, Boyd, Danah, & Davis, Marc. 2006. HT06, tagging paper, taxonomy, Flickr, academic article, to read. *HYPERTEXT '06: Proceedings of the seventeenth conference on Hypertext and hypermedia*, 31–40.

Riehle, D. 2008. How and Why Wikipedia Works: An Interview with Angela Beesley, Elisabeth Bauer, and Kizu Naoko. *Workshop Wikisym 2008*.

Rinard, Martin. 2003. Acceptability-oriented computing. *OOPSLA '03: Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, 221–239.

Russell, N., ter Hofstede, A.H.M., & van der Aalst, W.M.P. 2008. newYAWL: Designing a Workflow System using Coloured Petri Nets. *Proceedings of the International Workshop on Petri Nets and Distributed Systems (PNDS'08)*.

Russell, Nick, van der Aalst, Wil M. P., & ter Hofstede, Arthur H.M. 2006. Exception Handling Patterns in Process-Aware Information Systems. *BPM Center Report BPM-06-04*.

Schmidt, D.C. 2006. Model-Driven Engineering. *IEEE Compute*, **39**(2).

Schonenberg, Helen, Mans, Ronny, Russell, Nick, Mulyar, Nataliya, & van der Aalst, Wil M. P. 2008. Process Flexibility: A Survey of Contemporary Approaches. *CIAO! / EOMAS*.

Shafii, Reza. 2008. *Kaizen, BPM, and Agile Methodologies*. http://www.oracle.com/technology/pub/articles/dev2arch/2008/05/kaizen-bpm-agile.html (accessed June 1, 2009).

Spek, Sander, Postma, Eric O., & van den Herik, H. Jaap. 2006. *Wikipedia: organisation from a bottom-up approach*.

Wal, Thomas Vander. 2007. *Folksonomy*. http://vanderwal.net/folksonomy.html (accessed June 1, 2009).

Weske, M., van der Aalst, W.M.P., & Verbeek, H.M.W. 2004. Advances in Business Process Management. *Special Issue of Data and Knowledge Engineering - Elsevier Science Publishers*, **50**.

Weske, Mathias. 2007. *Business Process Management: Concepts, Languages, Architectures*. First edn. Springer Verlag.

Wu, Harris, Zubair, Mohammad, & Maly, Kurt. 2006. Harvesting Social Knowledge from Folksonomies. *HYPERTEXT '06: Proceedings of the seventeenth conference on Hypertext and hypermedia*, 111–114.

Yoder, Joseph W., & Johnson, Ralph E. 2002. The Adaptive Object-Model Architectural Style. *Pages 3–27 of: WICSA 3: Proceedings of the IFIP 17th World Computer Congress - TC2 Stream / 3rd IEEE/IFIP Conference on Software Architecture*. Deventer, The Netherlands, The Netherlands: Kluwer, B.V.