A Collaborative Approach to Maturing Process-related Knowledge

Hans Friedrich Witschel[1], Bo Hu[1], Uwe V. Riss[1],
Barbara Thönssen[2], Roman Brun[2], Andreas Martin[2], Knut Hinkelmann[2]

[1]SAP AG, Dietmar-Hopp-Allee 16
69190 Walldorf, Germany
{Hans-Friedrich.Witschel, Bo01.Hu, Uwe.Riss}@sap.com
[2]University of Applied Sciences Northwestern Switzerland (FHNW),
Institut for Business Information Systems,
Riggenbachstr. 16, 4600 Olten, Switzerland
{Barbara.Thoenssen, Roman.Brun, Andreas.Martin, Knut.Hinkelmann}@fhnw.ch

Abstract We introduce a new approach supporting knowledge workers in sharing process-related knowledge. It is based on the insight that - while offering valuable context information - traditional business process modelling approaches are too rigid and inflexible to capture the actual way processes are executed. Therefore, business process models are made agile and open for changes during execution. To achieve this, the strict distinction between build time modelling and run time execution are softened and process activities are represented to the users in a way that allows for individual adaptations. That can be done by attaching resources, commenting on an issue or adding problems and solutions to an activity or process. In addition activities can be delegated or new (sub-)activities can be added. Thus, the model can adapt to the reality of actual process executions and valuable resources and experiences are proactively presented to users in the right context. A double-staged approach is chosen to apply the model in the real application scenario of a university.

1 Introduction

Agility has emerged as an important common characteristic of successful businesses of any size, who benefit from quick response to volatile markets and rapid changing user requirements. In this work, we inspect business agility through the apparatus of knowledge sharing. More specifically, we leverage the process-related knowledge, in terms of knowledge about processes (process knowledge) and knowledge needed in processes (functional knowledge), to increase the agility of organisations. As such knowledge is used and generated during work, its sharing and maturing has to be aligned with business processes that faithfully reflect an organisation's core and mission-critical activities. Businesses need to make sure that employees, participating in mission-critical activities, share the acquired process-related knowledge to keep established experience within the organisation and optimise its performance in the face of employee fluctuation.

In practice, we deal with process-related knowledge with the help of Business Process Management Systems (BPMS) and/or Workflow Management Systems (WfMS). BPMS mainly manipulate process knowledge on the business level by making process structures and resources explicit and by supporting process improvement. But they lack process automation. This is the function of WfMS, which automate process execution (see [28], pp 8f). WfMS's modelling functionality, however, is specialised for technical aspects and is not rich enough regarding knowledge aspects. When aligned with everyday work activities, however, existing business process modelling and execution approaches may find themselves overstretched in answering the call of agility due to the lack of flexibility and the amount of overhead required for predefined process models. Usually, process models are created by experts who attempt to bring together all relevant knowledge about a certain process and model it in a BPMS. After negotiating and compromising, a resultant process model could truly represent how a process appears under certain generalised circumstances. The model, however, often differs from the reality of process execution [16]. Variations in execution, which are seldom readily documented, become inevitable when applying process models to new situations. The problem cannot be simply remedied with business process reengineering, which is carried out in a structured and systematic way and cannot keep pace with rapidly changing businesses or markets.

Coming from a totally different perspective, knowledge sharing (for all kinds of knowledge) sometimes takes place informally, e.g., via email and telephone or by imitation (apprenticeship). Although this is flexible and can be very efficient at times, it usually restricts the benefits to the persons that are directly taking part in the exchange (i.e., the information is completely lost for all others). Even if employees have documented their experience and made it publicly available (e.g. in a company Wiki, a lessons-learned database, or on a file share), it does not mean that others are aware of the existence of such information. Needless to say there is much less chance that they will be able to find it or that they will even look for it in a given work situation where it is needed.

The intrinsic inadequacy of formal and informal process knowledge sharing inspired us to take an eclectic approach so as to bridge exactly this gap: to learn process models by doing and to enable adding and sharing individual knowledge and experience. This approach enhances agile process modelling [11] with functional knowledge used in a specific process instance and the possibility of its informal exchange through so-called task patterns, during the execution of processes.

Our vision emphasises the participation of users in a succession of phases known as seeding, evolutionary growth and re-seeding in the SER model [5] (originally applied in the area of managing complex design environments). The SER model describes an approach "between the two extremes of 'put-all-the-knowledge-in-at-the-beginning' and 'just-provide-an-empty-framework' ". It combines the strengths and avoids the weaknesses of both top-down and bottom-up approaches, respectively. The SER model assumes that once a seed is taken up by a community, there is a phase in which the knowledge artefacts evolve in a ra-

ther uncontrolled way. According to Fischer, it is necessary not to force users to invest much effort into formalising their contributions since this would interrupt their normal work process (something most people are not prepared to accept). Contribution should be kept simple and will eventually lead to structures that are too redundant and unwieldy to be understood and managed. They are thus pruned and restructured in the reseeding phase, which is done by a knowledge engineer, removes inconsistencies and creates generalisations (i.e. removing pieces of information that are too context-specific) and formalisations of the knowledge. This is exactly what we want to achieve with the task pattern approach [22] that we introduce in this paper as mediator between process modelling and individual task execution. Projected onto the SER model, reseeding in our work is understood as a chance to understand and align the most frequent (and hence possibly most important) contributions to task patterns in order to learn about potential improvements of the original seeds. This realises a continuous improvement of process models and the task patterns based on actual work activities.

The paper is organised as follows: in Section 2 task patterns as the central building block for learning and maturing process-related knowledge is described. This is followed by a description of our approach to monitor performed tasks in order to semi-automatically support process model adaptations. In Section 3 we give an example of application of our new approach. Next, we describe some technical details of the system (Section 4), then give a brief overview on related work (Section 5) before Section 6 concludes.

2 Combining knowledge intensive processes with task patterns

A business process is a collection of structured activities with a precise goal to be achieved over a period of time. In general, the activities of a process are in a pre-defined order, resources are mapped (e.g. software systems or personnel, via roles) and the process flow is depending on fixed decision rules. The KISS approach [4] aims to bridge the gap of a strict distinction between design time and run time. A knowledge intensive process (KIP) can be regarded as a collection of activities building the 'skeleton' of a business process, some activities of which can be knowledge intensive (called 'KIA'). Whereas ordinary activities are always executed (i.e., in every process instanc), KIAs are optionally executed depending on information specific for the certain process instance. That can be application data, process data or functional data.

KIAs are modelled during build time but their execution is triggered – or suggested – during run time based on rules. If, for example, an application has to be checked, several KIAs could be executed such as 'Refer to an expert', 'Ask for additional material' or 'Clarify with applicant'. Which one is selected within a specific process instance depends on rules operating on run time information: information already provided for the application, decisions taken in previous process instances or data that is available from related information sources, e.g. out of a legacy system maintaining data of former applications.

We will call the concrete instance of an activity (assigned to particular members of an organisation) task, regardless whether it is a KIA or not. A task is a definition of a particular item of work that specifies the requirements and the goal of this work (cf. [1]). We introduce task patterns as abstractions of tasks that provide information and experience that is generally relevant for the task execution. By abstraction we mean common features of a family of similar tasks, which aim at the same goals under similar conditions (for details refer to [3,27]).

In this section, we describe how agile business processes can work together with task patterns to yield a new form of knowledge sharing. Meanwhile, in order to fully understand the way informal process knowledge to be attached to agile business processes, we explain the notion of task patterns more closely.

2.1 Task patterns

In our approach of maturing process knowledge described below, we will introduce a one-to-one relationship between an activity and a task pattern. That is, for each activity of an agile business process there is exactly one task pattern that serves as the basis for collecting information and experience around the tasks. Tasks concretise task patterns and thus instantiate the corresponding activity.

How does this facilitate the transfer of information and experience work? In general, task patterns provide two means for knowledge sharing (cf. [21,27]):

1. Abstraction services: these provide contextual information about resources that can be used in the task – including information objects such as files, but also persons who are to be contacted – or sub-tasks that should be started.
2. Problem/solution objects: These enable users to share experience regarding typical problems that may arise during the execution of a task, together with descriptions of possible solutions.

Users can interact with task patterns in two ways:

Consuming information from task patterns: when working on a task T, a suitable task pattern P can be displayed alongside the task. The user can access P's abstraction services to consume the resources that they offer and attach them to the current task T. The same applies to solutions offered for a problem that happens to occur in T.

Contributing to task patterns: users can also attach resources to their concrete task T while working on it. This enables them to associate such information with abstraction services or problem/solution objects of the task pattern P and publishing that information to a shared repository.

The approach maintains a clear separation between personal knowledge, contained in the individual task, and public experience, contained in the task pattern. This separation prevents an intermixture of private and public data. Therefore, it makes transparent to the user which task information is exposed and shared with others and which remains under individual control [20].

2.2 Sharing work experience

Regarding knowledge sharing, each activity of the process model is the basic unit to which information and experience gets attached. This happens via a one-to-one relationship between task patterns and activities (which are instantiated by tasks): for each activity, the corresponding task pattern collects the information and experience that users have attached to it while they were working on a task.

In the following, we will describe how agile business processes and task patterns play together. We differentiate such interplay into one that happens at design time and one that occurs during run time. At design time, for each activity, a draft of a corresponding task pattern is created with the aid of a group of experts who define an initial set of abstraction services and known problems together with their solutions. The initial pattern should be thought of as a seed that triggers the process of attaching experience to a work context. Subject to further refinement, the initial task patterns do not have to be (and will never be) complete in the beginning. These initial patterns are meant to grow larger while their users learn more about the activities and mature over time adapting to the actual way in which they are executed in practice.

All process information is stored in an ontology-based data store. Specifically, this store contains all information on task instances, encapsulated in so-called task description objects (TDOs). TDOs are filled dynamically by the back-end system and loaded into the front-end once a task gets accepted by a user (see below) - they thus serve as a means of communication between the two.

At run time, when the process is actually executed, the existing task pattern for an activity, which we denote by P, will be retrieved. A task pattern management system will first consult the task ontology to retrieve the abstraction services of P. It then helps users to instantiate P by recommending candidate fillers or each abstraction service. More specifically, one proceeds as follows (this flow of action is also depicted in Fig 1):

1. The workflow engine identifies which task T with corresponding task pattern P should be performed next.
2. A task T is instantiated and a set of organisational members is selected as potential executors of T. The selected persons are notified. They can accept or reject the request to execute T.
3. Since T is the instantiation of an activity A and since for each activity A, there is exactly one task pattern P assigned to it, the workflow engine will next retrieve that task pattern P.
4. It then determines the context of T and uses it to retrieve relevant resources that should be added to abstractor services of the task pattern P. These resources will be added to the TDO corresponding to T.
5. Once the first person has accepted the task, the corresponding TDO is fetched, together with the task pattern P. P is enriched with the information in TDO and both the task details and the task pattern are displayed to the user in a task management application.
6. The user can then start working on the task, making use of the information provided in the task pattern P. Resources, but also problem/solution

objects can be easily copied from the task pattern into the task, becoming attachments to it.

7. On the other hand, the user can also attach her own resources (that she considers useful in the context of T) to T.
8. Enhancements of a task pattern are stored locally. They will be available if the same user performs another task T' that corresponds to the same A (i.e. when the task pattern is loaded next time). However, if the user chooses to publish the enhancements, she can do so by a simple click, making them available to the others who have to perform a task of type A.
9. When the user has finished working on the task, she sets the status to "completed" and the workflow engine is notified of this.

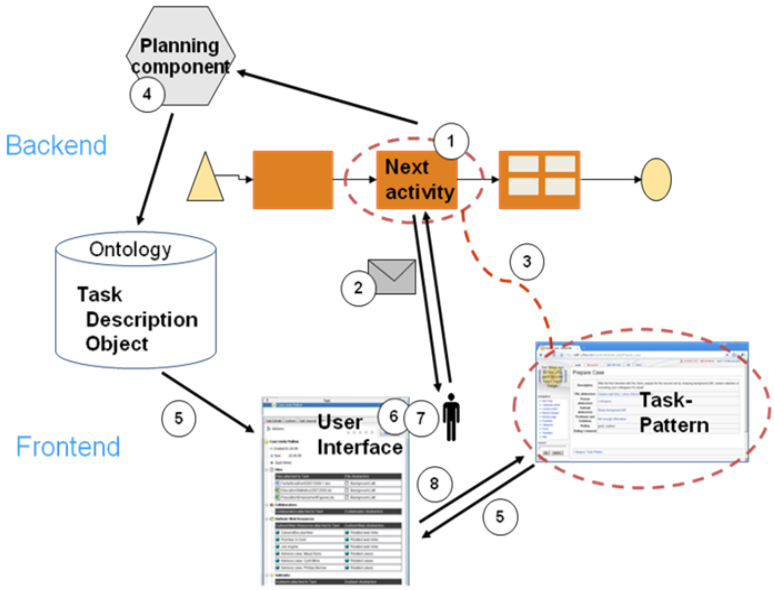

Figure 1. Flow of action during runtime

2.3 Learning from work experience

As mentioned in the introduction task patterns should be thought of as seeds to aggregate information contributed by end-users. This participation becomes particularly relevant since we do not believe in fully automatic improvements of process models. Instead we envisage a tool that is able to detect deviations and analyse the collected data in order to make suggestions for process changes, in a way similar to work carried out in the area of process mining, e.g. [30]. Based on

those blueprints the knowledge engineer will be able to decide on the suggested changes. For process improvement the following aspects will be analysed [3,27]:

Subtasks that are frequently added to a task (or as subtask abstractor to the corresponding task pattern): if many users add the same (kind of) subtask to a given task, this indicates that potentially the process model can be improved by including that subtask as a new activity.

Delegation of tasks can indicate that either the work balance is not correctly considered or the skills of the assigned persons are not appropriately evaluated. Rules for resource allocation should be adapted accordingly.

Problem/solution objects added to a task pattern can be included by other users in their tasks. If many users do so, it means that the problem occurs frequently and that it should be considered for process or task pattern improvements.

Resources such as documents or persons can be attached to abstractor services of task patterns, indicating the contexts in which they are useful (namely the process, and activity, task, respectively they are being used in). An analysis of these contexts is generally of interest as it may help to categorise the resources according to their domains of application. Similarly, an analysis of the set of all documents attached to task patterns can lead to a categorisation of documents based on the type of situation(s) in which they are consumed.

In all these cases, an initial step in the analysis is aligning the corresponding items with each other, e.g. to find out that two problem descriptions refer to the same (type of) problem in reality.

3 An application scenario

This section presents the scenario of an evaluation study that was performed within the University of Applied Sciences Northwestern Switzerland (FHNW) in the context of the EU-funded project MATURE to elicit application scenarios and requirements for the ICT system that is being built to realise the process-related knowledge maturing concepts presented in this paper. Therefore, a prototype (later also called "demonstrator") was built that implements the concepts described here.

The model for the business process of matriculation is shown in Fig 2. We can see that the student, the administration office and the dean are involved in the process; tasks can be assigned to the administration office or to the dean as they directly interact with the system. KIAs are highlighted.

The matriculation process starts with a student's application request. After the receipt of the request, several checks of the application have to be executed in a KIP. As it is shown, the KIAs will not be executed in a pre-defined order. The reason is the following: Depending on where the applicant comes from (but also further criteria), different activities have to be performed. E.g. the availability of a matriculation number has only to be checked if the applicant is

from Switzerland. Therefore, a *variable process identification and selection service* automatically chooses the needed activities and assigns them to the possible executor of the activity. The determination of study fees is based on given regulations and can be supported using a *constraint checking service* for decision making. Further on, a *resource allocation service* assigns artefacts based on given criteria. For instance, when checking the approval of a university, appropriate websites or experts from a respective nation are attached to the activity.

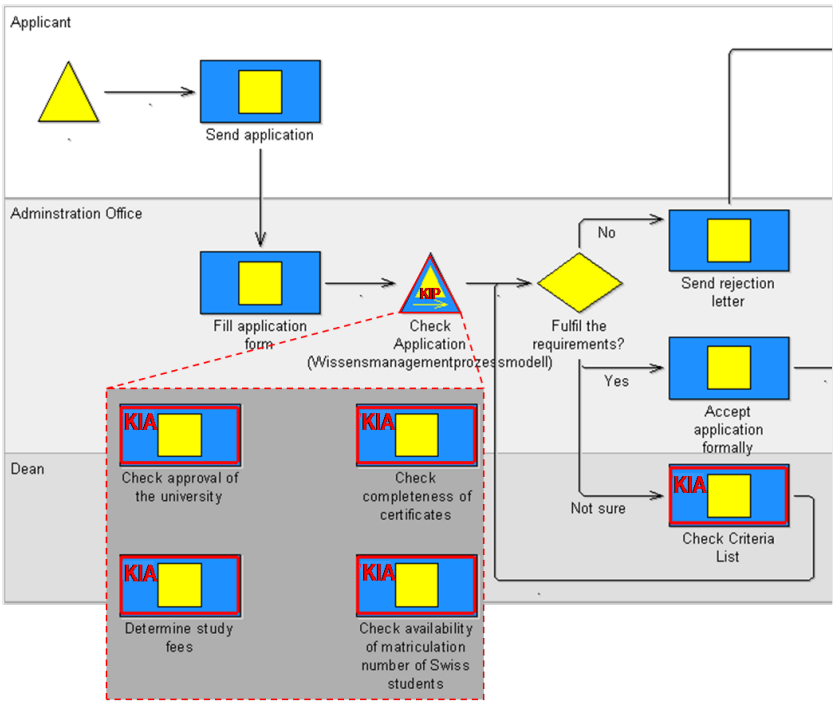

Figure 2. First part of the matriculation process model and the KIP sub-process "Check application".

After these checks, the process goes on and it is decided whether the needed requirements are fulfilled or not. The *branching and decision making service* can be invoked in order to decide, whether it is already clear at this stage that the requirements cannot be fulfilled by the applicant and a rejection letter has to be sent. Otherwise the applicant is invited to an interview. Afterwards an interview will be held, the application dossier will be updated and a commission meeting will be held to decide about the acceptance or rejection of the applicant. The process continues with mainly administrative activities until it reaches the end.

3.1 Example of task pattern application

Now, we illustrate the application of task patterns by giving an example from the matriculation scenario described above. Let us consider the task of checking the completeness of an applicant's certificates (part of the knowledge intensive sub-process "Check Application" displayed in Fig 2).

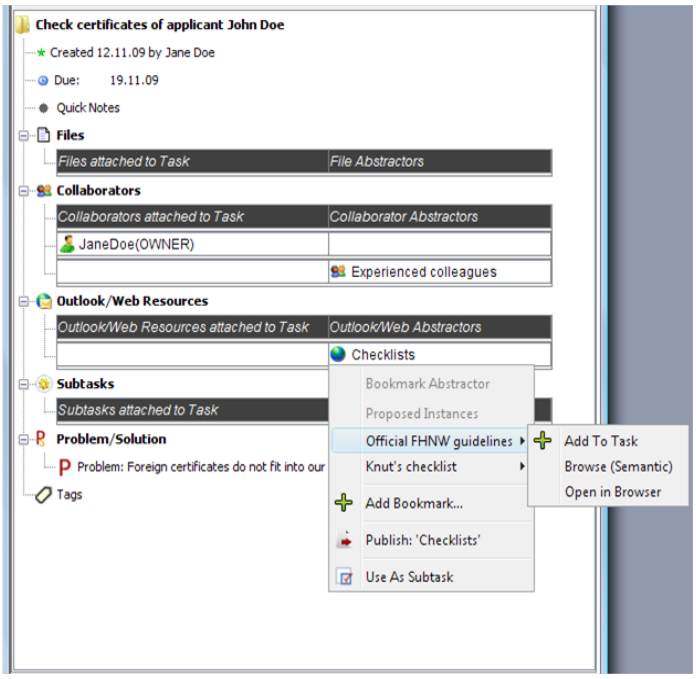

Figure 3. An example of a task pattern and how to consume information from it.

Fig 3 shows a task pattern that corresponds to this activity. More precisely, the details of the current task - namely "Check certificates of applicant John Doe" - are displayed (e.g. due date and owner of the task) on the left-hand side. The task pattern with its abstractor services (called "abstractors" in the UI) is displayed on the right-hand side. We can see two abstractor services: i) experienced colleagues: colleagues who have handled many applications; ii) checklists: lists that provide guidance as to what should be checked and how.

The figure also shows a context menu that appears when right-clicking on the latter abstractor service. It contains the resources that are offered by the abstractor service. Clicking on "Add to Task" will result in a resource being displayed on the left-hand-side, vertically aligned with the corresponding abstractor service. Imagine, for example, that a new colleague by the name of Jane has to work on the task at hand and needs to get acquainted with the official guidelines.

In that case, she would access the information in the abstractor service "Checklists" and consult the "Official FHNW guidelines" as depicted in the figure. A similar context menu exists for resources on the left-hand side, allowing these to be added to abstractor services and thus for the contribution of resources to task patterns. For example, Jane might discover – after having worked for some time on various student applications – that a few additional things usually need to be checked, which she documents in her own private checklist that she attaches to her tasks. Via the context menu, she can add this checklist to the "Checklist" abstractor service such that it becomes available for other users. Thus, using these context menus, end-users can easily consume information offered in task patterns and contribute to them.

3.2 Demonstrator evaluation

In order to verify the automated process knowledge maturing, comprehensive use of the approach is necessary. Therefore an evaluation phase of one month where the demonstrator was used, including intermediate and post-evaluation interviews, was conducted.

The main aim of the evaluation was to use the demonstrator in a productive environment. Therefore it was of special interest whether KISSmir addresses the clear need and/or problem in the tester's context. The post-interview should give an insight about the use of all functionalities, suggestions, tasks patterns, knowledge sharing but also technology acceptance. Of further interest was the perceived degree of support of the demonstrator regarding maturing of process-related knowledge and how it could be improved.

The results showed first that process adaptations do actually surface as a result of the use of the demonstrator. Although the matriculation process was modeled together with the end users, some adaptations of it were detected to be necessary during productive use. Furthermore necessary changes in rules were identified. As a second point, the process support was experienced as big benefit of using the demonstrator. The end users liked to be reminded about the tasks needed to be executed and being guided by the sequence. As the number of tasks and their sequence varies for each process execution, they didn't have to think about which tasks need to be done and were able to accomplish the tasks more quickly. Thirdly, the resource recommender functionality was analysed. For each task, experts, historical cases and web-links were proposed (if available) by the demonstrator. The post-evaluation interview showed that with exception of web-links these suggestions have been used very seldom. The main reason for the limited use was mainly the unawareness of it and a (too) small knowledge base. Further features as quick notes which can be added to any task at any time and problem/solution objects were perceived as being useful. Last but not least the knowledge sharing functionalities were analysed. By adding resources to task patterns and publish this information, knowledge can be shared. The same can be done with problem/solution descriptions. However, similar as for the suggestions, the interviewed persons think these are nice functionalities as needed information could be found easier and faster, but did not use it extensively.

4 Implementation

4.1 Process modelling and execution

The modelling of an adaptive business process can be performed in a semantic modelling environment like ATHENE [9] or WSMO-Studio [2]. ATHENE provides the creation of several models (process model, organisational model, etc.) based on ontologies. The activities of the processes will be linked with the related task patterns and resources (files, roles, etc.) which are stored in semantic repositories (❶ in Fig 4). Further on, the process model can be enhanced with adaptivity services (❷). After modelling the process in ATHENE it can be transferred to the execution framework and stored in a semantic repository (❸). The model is represented in a knowledge representation language like RDF/S[1] or OWL[2] and will be transformed via a transformation service into a process execution language like BPEL[3] or XPDL[4].

The transformed process can be executed in the execution framework (e.g., BPEL workflow engine) (❹). The process can access the linked resources which are stored in the semantic repositories. During run time the process can invoke the defined adaptivity services. The "task management service", which is part of the process framework, invokes the task GUI (graphical user interface) (❺). The instance management service stores and holds the instances.

4.2 Task pattern and task management

Tasks and task patterns are delivered to the user through a Personal Task Management infrastructure. That infrastructure is part of the NEPOMUK Social Semantic Desktop [7].

It consists of a semantic task management framework (STMF [19]) which offers task-related (web) services over the entire desktop and handles the manipulation, storage and retrieval of all task and task pattern-related information; information is stored locally in the RDF repository of the Social Semantic Desktop in a way that ensures seamless semantic integration of information objects and task representations.

As a user interface, the KASIMIR sidebar [6] has been developed, which builds on the STMF task services and makes Task Management functionality available to end- users. KASIMIR allows users to assign basic task properties and to attach involved persons, information objects and subtasks. It also allows users to view task patterns attached to a task, consume its resources and contribute In addition to the local storage of personal task (pattern) information, there is a server component that stores public task patterns. It is based on Semantic MediaWiki (SMW[5]), which allows the initial modelling and later adaptation

[1] http://www.w3.org/TR/rdf-schema/
[2] http://www.w3.org/TR/owl-features/
[3] http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
[4] http://www.wfmc.org/xpdl.html
[5] http://semantic-mediawiki.org/

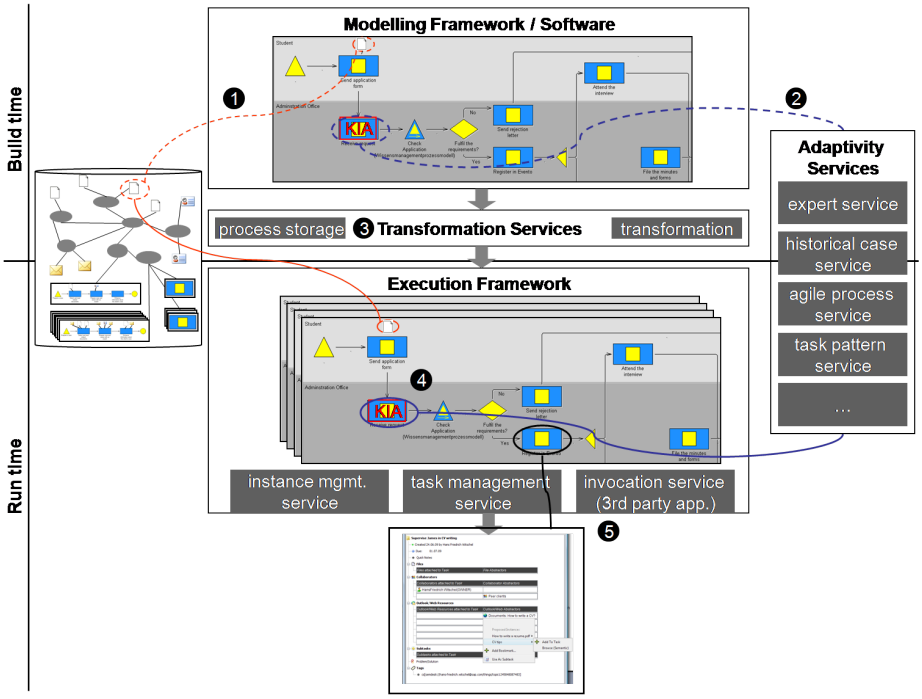

Figure 4. Customer Information Portal Architecture

of public task patterns over the Wiki's user interface. Entities related to task patterns are modelled as SMW semantic templates.

5 Related Work

Making business processes agile to meet the requirements of knowledge intensive work and faster changing business environments is a topic that has already been addressed for some time. It is guided by the insight that in knowledge intensive processes the particular sequence of tasks is often variable and depends on the information at hand. Traditionally Workflow Management Systems (WfMS) distinguish between design time and run time [12] and it is the dependency of the process on input information that makes this distinction to become blurry. An overview of approaches to tackle this problem can be found in [22].

Usually a process model containing all activities and resources is created during design time. Flexibility is provided through modelling choices and merge-constructs. This can lead to highly complex models which are hard to maintain [25]. In addition, especially in the tertiary sector the processes are knowledge-intensive and cannot be foreseen for all exceptional situations and circumstances. During run time exceptional situations, unforeseeable events and unpredictable

situations have to be dealt with. Therefore van der Aalst et al. introduced case handling 'as a new paradigm for supporting flexible business processes' [31] in order to avoid predefined process execution.

However, supporting flexible process execution does not cover all of the dimensions of change in business processes (dynamism, adaptability, flexibility) as introduced by Sadiq et al. [25]. For this, tracking and mining of the actual process/task variations users perform are necessary.

To support agility, semantic technologies have been used in several approaches, amongst others by [8] for process implementation and querying by [14] to build their 'agent based business process management system' or by [22] to facilitate task patterns. Especially the pattern approach is tightly built on semantic technologies and an approach for combining it with Service-Oriented Architecture has been proposed [23]. Another semantic approach to process management based on Unified Activity Management has been suggested by Moran et al. [18]. A general overview is given by [17]. Recently, Feldkamp, Hinkelmann et al. introduced the 'KISS approach' combining semantically enriched process models with business rules ([18], [10]). Although this approach reaches the flexibility to execute agile processes, two aspects are not yet covered: a) how to share the knowledge gained through task handling without cumbersome publishing and b) how to automatically detect execution variances. In this paper we have shown how deviations between the actual process execution and the process model can be identified and how adaptations can be recommended automatically. Question a) is addressed in section 2.2, whereas question b) is detailed in section 2.3.

As far as the world of business process modelling is concerned, approaches to collect and mature process knowledge collaboratively are scarce. [15] has proposed an architecture that integrates knowledge management and business process management. However, this approach follows a traditional expert-driven way. Approaches in the field of process mining (e.g. [31]) - which try to extract process knowledge from implicit information contained in system event logs - exploit user interaction, but do not actually encourage explicit user contributions to an evolving process knowledge repository. With respect to knowledge work this is a complicated task since the nature of individual tasks and the associated experience cannot be identified properly enough to enable successful knowledge proliferation. Sharing process knowledge in a task management environment has been explored, for example, in [13], suggesting to copy information from previous related tasks. Task patterns, as a more elaborate way of mediating experience transfer have been proposed in [22] and elaborated further, cf. e.g. [3,20,27]. In [29], a more process-oriented view of task patterns has been introduced where users can exchange and collaboratively develop lightweight process models.

6 Conclusion and Future Work

In this article, we have outlined a new paradigm of knowledge and experience sharing that enhances agile business processes. This is done through connecting activities in formal process models with loosely regulated task patterns emerged

from our everyday work. The former gives guidance to the business target while the latter allows us to proceed in a way that best suits the end-users' needs. Task patterns capture how people carry out a task (process knowledge) and how people leverage resources in supporting their solutions (functional knowledge). They thus help to avoid the problem of rigidity inherent in traditional process modelling approaches since it involves the end-users in shaping the support that the model offers and since it eventually adapts to the reality of end-users' process execution.

The results of our evaluation of the prototype with the University of Applied Sciences Northwestern Switzerland are promising: both the conceptual framework and the prototype were well accepted even though some non-technical barriers were identified.

In our approach we have followed the idea of knowledge maturing as a process that understands "[...]learning activities as embedded into, interwoven with, and even indistinguishable from everyday work processes[...]" [26]. According to the knowledge maturing concept learning is seen as a social and collaborative activity, in which individual and organisational learning processes are dynamically interlinked among each other [24]. This approach has been applied to knowledge intensive processes where the continuous collaborative enhancement appears as particularly important due to continuously changing work targets and situations.

Furthermore, we envisage the following improvement to our approach. For the future, it is planned to implement and deploy the agile business process (together with appropriate task patterns) at the project application partners and to observe if and how the intended process knowledge maturing takes place. Compared to other types of knowledge, the evolution of process knowledge is less transparent and thus more difficult to analyse. The interplay between task patterns and process model provides valuable insights. Indeed, after having the process productive over a certain period, a reasonable amount of real-life usage data of tasks and task patterns can be accumulated. Such data provide the ground for automatic or customised business process model updates.

Acknowledgements

This work is supported by the European Union IST fund through the EU FP7 MATURE Integrating Project (Grant No. 216356).

References

1. Katriina Byström and Preben Hansen. Conceptual framework for tasks in information studies. *Journal of the American Society for Information Science and Technology*, 56(10):1050–1061, 2005.
2. Marin Dimitrov, Alex Simov, Vassil Momtchev, and Mihail Konstantinov. Wsmo studio—a semantic web services modelling environment for wsmo. In Enrico Franconi, Michael Kifer, and Wolfgang May, editors, *The Semantic Web: Research and Applications*, volume 4519 of *Lecture Notes in Computer Science*, chapter 53, pages 749–758. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

3. Ying Du, Uwe V. Riss, Ernie Ong, Liming Chen, David Patterson, and Hui Wang. Work experience reuse in pattern based task management. In *9th International Conference on Knowledge Management (I-KNOW)*, pages 149–158, 2009.
4. Daniela Feldkamp, Knut Hinkelmann, and Barbara Thönssen. Kiss: Knowledge-intensive service support: An approach for agile process management. In *Advances in Rule Interchange and Applications, International Symposium (RuleML)*, pages 25–38. 2007.
5. Gerhard Fischer, Jonathan Grudin, Raymond McCall, Jonathan Ostwald, David Redmiles, Brent Reeves, and Frank Shipman. Seeding, evolutionary growth and reseeding: The incremental development of collaborative design environments, 2001.
6. Olaf Grebner, Ernie Ong, and Uwe V. Riss. Kasimir: Work process embedded task management leveraging the semantic desktop. In *Multikonferenz Wirtschaftsinformatik (MKWI 2008)*, pages 715–726, Berlin, 2008. GITO-Verlag.
7. Tudor Groza, Siegfried Handschuh, Knud Möller, Enrico Minack, Mehdi Jazayeri, Cédric Mesnage, Gerald Reif, and Rósa Gudjónsdóttir. The nepomuk project- on the way to the social semantic desktop, 2007.
8. Martin Hepp, Frank Leymann, John Domingue, Chris Bussler, Alexander Wahler, and Dieter Fensel. Semantic business process management: A vision towards using semantic web services for business process management. In *IEEE International Conference on e-Business Engineering (ICEBE)*, pages 535–540, 2005.
9. Knut Hinkelmann, Simon Nikles, and Lukas von Arx. An ontology-based modeling tool for knowledgeintensive services. In *1st International Conference on Methodologies, Technologies and Tools Enabling E-Government*, pages 43–56, 2007.
10. Knut Hinkelmann, Fabian Probst, and Barbara Thönssen. Agile process management framework and methodology. In *AAAI Spring Symposium on Semantic Web Meets e-Government*, 2006.
11. Knut Hinkelmann, Barbara Thönssen, and Fabian Probst. Referenzmodellierung für e-government-services. *Wirtschaftsinformatik*, 5:356–366, 2005.
12. David Hollingsworth. The workflow reference model. workflow management coalition, 1993.
13. Harald Holz, Oleg Rostanin, Andreas Dengel, Takeshi Suzuki, Kaoru Maeda, and Katsumi Kanasaki. Task-based process know-how reuse and proactive information delivery in tasknavigator. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 522–531, New York, NY, USA, 2006. ACM.
14. N. R. Jennings, T. J. Norman, P. Faratin, P. O'Brien, and B. Odgers. Autonomous agents for business process management. *Applied Artificial Intelligence: An International Journal*, 14(2):145–189, 2000.
15. J. Jung, I. Choi, and M. Song. An integration architecture for knowledge management systems and business process management systems. *Computers in Industry*, 58(1):21–34, January 2007.
16. D. Karagiannis, S. Junginger, and R. Strobl. Introduction to business process management systems. In B. Scholz-Reiter and Eberhard Stickel, editors, *Business process modelling*, pages 81–106. Springer, 1996.
17. Florian Lautenbacher and Bernhard Bauer. A survey on workflow annotation & composition approaches. In *Workshop on Semantics for Business Process Management (SBPM)*, 2007.
18. Thomas P. Moran, Alex Cozzi, and Stephen P. Farrell. Unified activity management: supporting people in e-business. *Commun. ACM*, 48(12):67–70, 2005.

19. Ernie Ong, Olaf Grebner, and Uwe V. Riss. Pattern-based task management: Pattern lifecycle and knowledge management. In *4th Conference of Professional Knowledge Management (WM 2007)*, volume 2, pages 357–364, 2007.
20. Uwe V. Riss, Ulrike Cress, Joachim Kimmerle, and Stefan Martin. Knowledge transfer by sharing task templates: two approaches and their psychological requirements. *Knowledge Management Research & Practice*, 5(4):287–296, 2007.
21. Uwe V. Riss, Olaf Grebner, and Ying Du. Task journals as means to describe temporal task aspects for reuse in task patterns. In *9th European Conference on Knowledge Management*, pages 721–730, 2008.
22. Uwe V. Riss, Alan Rickayzen, Heiko Maus, and Wil M. P. van der Aalst. Challenges for business process and task management. *Journal of Universal Knowledge Management*, 0(2):77–100, 2005.
23. Uwe V. Riss, Ingo Weber, and Olaf Grebner. Business process modelling, task management, and the semantic link. In *AAAI Spring Symposium AI Meets Business Rules and Process Management*, pages 99–104, 2009.
24. Uwe V. Riss, Hans F. Witschel, Roman Brun, and Barbara Thönssen. What is organizational knowledge maturing and how can it be assessed? In *9th International Conference on Knowledge Management (I-KNOW)*, pages 28–38, 2009.
25. Shazia Sadiq, Wasim Sadiq, and Maria Orlowska. Pockets of flexibility in workflow specification. In Kunii, Sushil Jajodia, and Arne Sølvberg, editors, *Conceptual Modeling ER 2001*, volume 2224 of *Lecture Notes in Computer Science*, chapter 38, pages 513–526. Springer Berlin Heidelberg, Berlin, Heidelberg, December 2001.
26. Andreas Schmidt, Knut Hinkelmann, Tobias Ley, Stefanie N. Lindstaedt, Ronald Maier, and Uwe Riss. Conceptual foundations for a service-oriented knowledge and learning architecture: Supporting content, process and ontology maturing. In Tassilo Pellegrini, Sóren Auer, Klaus Tochtermann, and Sebastian Schaffert, editors, *Networked Knowledge - Networked Media*, volume 221, chapter 6, pages 79–94. Springer Berlin Heidelberg, 2009.
27. Benedikt Schmidt and Uwe V. Riss. Task patterns as means to experience sharing. In Marc Spaniol, Qing Li, Ralf Klamma, and Rynson W. H. Lau, editors, *Advances in Web Based Learning - ICWL 2009*, volume 5686 of *LNCS*, chapter 42, pages 353–362. Springer, Berlin, Heidelberg, 2009.
28. Workflow Management Coalition Specification. *Workflow Management Coalition, Terminology & Glossary (Document No. WFMC-TC-1011)*. Workflow Management Coalition Specification, February 1999.
29. Todor Stoitsev, Stefan Scheidl, and Michael Spahn. A framework for light-weight composition and management of ad-hoc business processes. pages 213–226. 2007.
30. Wil van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.
31. Wil M. P. van der Aalst, M. Weske, and D. Grünbauer. Case handling: a new paradigm for business process support. *Data & Knowledge Engineering*, 53(2):129–162, May 2005.