A pattern repository for establishing inter-organizational business processes

Document status and date:
Published: 01/01/2006

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

A Pattern Repository for Establishing Inter-Organizational Business Processes

A. Norta, M. Hendrix, P. Grefen

Eindhoven University of Technology, Faculty of Technology Management, Department of Information Systems, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands.
a.norta@tm.tue.nl

Abstract. In the domain of business-to-business (B2B) collaboration, companies are pursuing the objective of electronically linking their business processes for improving their supply chains. For creating such inter-organizational collaboration, intra- and inter-organizational knowledge workers (IKWs) function as assisting experts. However, IKWs must not constantly "reinvent the wheel" but should instead be supported by a repository that contains knowledge about how to design business processes. Thus, this paper proposes the support of IKWs by a pattern repository for the effective and efficient design of inter-organizational business processes. A pattern is conceptually formulated knowledge that is technology independent. By storing patterns in a uniform specification template of a meta model, it is possible to perform systematic reasoning. Having information readily available about the technology support of individual patterns, IKWs can quickly analyse with which intersection of pattern sets it is possible to link intra-organizational business processes.

1 Introduction

Companies that focus on their core competencies or miss know-how to perform certain business activities source services from providers. In this context, a service consists of a business process that is integrated into the in-house process of the consuming company. For example, a truck-producing company has suppliers of a water tank, an insurance company uses a third party to assess damage cases. A promising approach for B2B is the coupling of workflow concepts with service-oriented business integration. This emerging framework of dynamic inter-organizational business process management (DIBPM) [15] offers a new model for addressing the need of organizations for dynamically bringing together a service consumer and a service provider over web-based infrastructures where the service is a business process. To do so, DIBPM merges service-oriented business integration (SOBI) and workflow management concepts. The setup of such B2B commerce is a client-server relationship where one party offers a service that is integrated into the process of a consumer.

To establish intra- and inter-organizational business processes efficiently and effectively in DIBPM, the utilization of patterns is recommendable. Corporations typically comprise an information infrastructure consisting of a heterogenous system environment supporting their business processes. The situation turns even more complex when the business processes of collaborating parties are linked. By checking which patterns

the respective heterogenous system environments support, a common denominator of collaboration is detected. Control-flow patterns [6–8] have been specified after investigating several intra-organizational workflow management systems. Furthermore, patterns for intra-organizational data-flow and resource management [22, 23] have been discovered and specified. More recently so-called service-interaction patterns [9] have been specified for the coordination of collaborating processes that are distributed in different, combined web services.

In the domain of SOBI, web service composition languages (WSCL) have emerged for supporting process specifications, e.g., BPEL, BPML [10, 11] and so on. Such languages compose services in a workflow, offering a complex service that carries out activities. The referenced pattern specifications and emerged WSCLs show that a rich amount of results exist that are relevant for DIBPM. For example, many e-business related patterns are textually available online [3] for the perspectives business-, integration-, composite-, custom design-, application- and runtime patterns. For inter- and intra-organizational knowledge workers (IKWs) who are exposed to business, technological, and conceptual complexity, such patterns promise a meaningful support for effectively and efficiently establishing inter-organizational business processes with the help of SOBI technology. IKWs organize the business processes in-house and establish business process links for B2B activities. They manage the heterogenous system infrastructure that supports such business processes. However, the pattern specifications of various perspectives that IKWs need to employ, differ and it has not been investigated how they relate to each other across different perspectives.

Originating from the cognitive sciences, e.g., philosophy, psychology, the Language-Action-Perspective (LAP) facilitate the construction of automated, coherent messaging between information systems, as has been observed in many research works [13, 17]. Briefly, LAP emphasizes what people *do* while communicating; how they create a common reality by means of language and how communication brings about a coordination of their activities. The approach of LAP is applicable in business collaboration [12] where inter-organizational transactions are intuitively modelled and carried out.

It is desirable to store all the pattern related data uniformly in one knowledge base and make it accessible for IKWs with tool support. Looking at the pattern repositories that are cited above, their content is always static and limited to either one or a couple of perspectives. However, for IKWs it is desirable to have a repository available that is interactive and dynamically growing in perspectives and content. The repository should store knowledge about how patterns relate to each other within the same perspective and across different perspectives. This paper fills the gap by presenting a pattern meta model that allows dynamic growth in content by permitting the admission of new patterns that may belong to newly introduced perspectives. Furthermore, a reference architecture is proposed for the development of tools that use the pattern meta model and that support IKWs in employing patterns for the creation of intra- and inter-organizational business processes.

The structure of this paper is as follows. First, Section 2 describes how IKWs are involved in realizing inter-organizational business process collaboration. Next, Section 3 gives an overview of a pattern meta model that is used for uniformly storing and relating patterns to each other. The subsections describe the meta model in detail. In Section 4,

the lifecycle of a pattern is used to deduct requirements a pattern repository must fulfil that runs on top of the pattern meta model. Section 5 presents related work and Section 6 concludes the paper.

2   Inter-Organizational Business Process Collaboration

This section gives a definition of DIBPM and relates inherent perspectives to each other. Furthermore, the nature of IKW involvement in DIBPM is made explicit. A definition of DIBPM [15] is given as follows: A dynamic inter-organizational business process is formed dynamically by the (automatic) integration of the subprocesses of the involved organizations. Here dynamically means that during process enactment collaborator organizations are found by searching business process market places and the subprocesses are integrated with the running process.

Important issues in connection with DIBPM are the definition and identification of processes, the way compatible business partners find each other efficiently, the dynamic establishment of inter-organizational processes, and the setup and coupling of inter-organizational processes for enactment. In Figure 1 different perspectives are depicted for creating an inter-organizational business process collaboration. To the left and right two factory symbols represent the collaborating organizations that have their internal legacy systems. Those legacy systems are linked with intra-organizational business processes that combine several perspectives. In Figure 1 the intra-organizational perspectives control-flow, resource, data-flow, and transaction are depicted. However, it is possible that further perspectives are included or omitted.
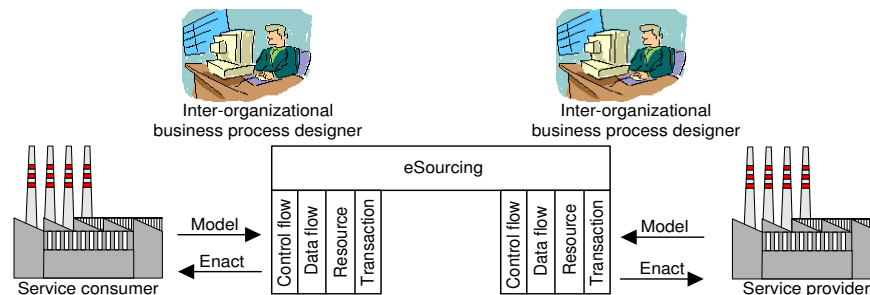
Fig. 1. Knowledge workers in inter-organizational collaboration

An additional perspective is contained in Figure 1 that crosses the boundaries of collaborating organizations, namely the eSourcing [19] perspective for which a catalogue of patterns [2, 20, 21] is specified. In the context of DIBPM, eSourcing is a framework for harmonizing on an external level the intra-organizational business processes of a service consuming and one or many service providing organizations into a B2B supply-chain collaboration. Important elements of eSourcing are the support of different visibility

levels of corporate process details for the collaborating counterpart and flexible mechanisms for service monitoring and information exchange. For sake of brevity in this paper, in [2], eSourcing interaction and construction patterns, eSourcing-configuration examples, and corresponding documentation are available online.

On top of Figure 1 two IKWs are depicted that each belong to a collaborating organization. Although the creation of eSourcing configurations should be carried out as automated as possible, it is realistic that IKWs remain provisionally necessary for the foreseeable future. For carrying out their work effectively and efficiently the support of a knowledge base is sensible that contains perspective-specific patterns for intra- and inter-organizational business process management. Thus, the next section presents the main building blocks of a meta model that is suitable for patterns of arbitrary perspectives.

3 The Pattern Meta Model

In the introduction of this paper several pattern sources of differing perspectives are referenced. The specifications of those patterns use templates with similar keywords. For example, the keyword `description` is generally used to specify the properties of a pattern. The keyword `example` is consistently used for listing real-world or abstract instances where the pattern occurs. However, many pattern specifications use a template of keywords that deviates or where the keywords are used differently. By harmonizing pattern-specification templates in a pattern meta model, IKWs can comprehend better the differences, commonalities, relationships of patterns.

The meta model must be able to fill in new patterns belonging to newly introduced perspectives that are relevant for DIBPM. It is predictable that the meta model will capture a considerable number of patterns, an IKW should be able to quickly find them based on characterizing search options. As inter-organizational business processes are supported by a heterogenous system environment, a meta model needs to capture technology support.

Such information is useful for IKWs to determine with which commonly supported patterns inter-organizational collaboration can be established. Finally, besides IKWs different types of users of a pattern meta model based repository are predictable such as an administrator, users who submit patterns, pattern reviewers, and so forth. A meta model must capture information of different user types for managing access rights to pattern information. Consequently the following subsection first presents the pattern meta model and their relationships as black boxes. Next, those packages are presented as white boxes that detail the relationship to elements belonging to other packages.

3.1 Meta-model Packages

The left side of Figure 2 depicts a model of packages that are related to each other. These packages encapsulate classes that are explained in following sections. The center of the package-model is named `Pattern`, which contains all classes that capture information for specifying a patterns. In the `Taxonomy` package, classes are contained that capture information about DIBPM perspectives. This package contains classes that create a

taxonomy into which patterns can be embedded. The `Support` package encapsulates classes for managing information about technologies that support patterns. Finally, the `User Management` package captures information of different users of the pattern repository, e.g., administrator, reviewer, pattern submitter, and so on.
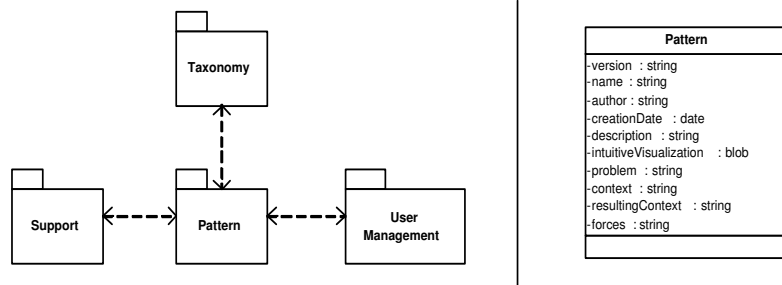

Fig. 2. Meta-model packages with their dependencies and the `Pattern` class

On the right side of Figure 2 the core class of the Pattern package is depicted, which is equally named `Pattern`. The attributes of this class form the main description template of a pattern specification. A pattern has a `version` and a `name` that should be meaningful. Furthermore, a pattern has an `author` and a `creationDate` for every version. The `description` of a pattern mentions the inherent pattern properties and describes the relationship between them. Furthermore, the `intuitiveVisualization` contains a model that helps to support the comprehensibility of the pattern description. The `problem` of a pattern is a statement describing the context of pattern application. In this context conflicting environmental objectives and their constraints are described. The application of a pattern in that context should result in an alignment of the given objectives. Next, the `context` states a precondition, which is the initial configuration of a system before the pattern is applied to it. On the other hand, the `resultingContext` describes the postcondition and possible side-effects of pattern application. Finally, the `forces` describe trade-offs, goals and constraints, motivating factors and concerns for pattern application that may prevent reaching the described postcondition.

After presenting an overview of the pattern meta model, the following subsections present detailed models of the packages depicted in Figure 2. The detailing classes of package `Pattern` are presented as a white box. When the packages `Taxonomy` and `Support` are presented in detail, their references to classes in the `Pattern` package are depicted explicitly. Since the package `UserManagement` is of relevance for an application running on top of the pattern meta model, the latter package is presented in Section 4 together with the functionality of the pattern repository. If required, attributes of classes are explained in further detail. However, for sake of brevity such attribute explanations are omitted in many cases.

3.2 The Pattern-Taxonomy Model

As the first package that is presented in detail, Figure 3 depicts the classes of the
`Taxonomy` package. Creating a taxonomy is relevant for ordering patterns and relating
them to each other. Additionally, a taxonomy helps to find patterns that are stored in the
repository.

Fig. 3. Detailed class model of the `Taxonomy` package

The introduction of this paper references patterns belonging to DIBPM-relevant per-
spectives, e.g., *control-flow*, *data-flow*, *resource*. Thus, in the depiction of package Tax-
onomy in Figure 3 the class named `Perspective` is central. Informally, a perspective
can be seen as a particular angle from which a certain domain is perceived. To the left
of Figure 3 the relationship to the `Pattern` package is depicted. It shows that a pattern
always only belongs to one perspective while a perspective references many patterns.

Publications of pattern specifications contain groupings of patterns that share char-
acteristics. For example, the *white-box*, *black-box*, and *grey-box* patterns [2, 21] of the
eSourcing perspective are in one group with the characteristic *contractual visibility*. If
a considerable amount of patterns is in the repository, a more detailed grouping of pat-
terns is sensible to allow IKWs speedy pattern discovery. Therefore, Figure 3 shows
a class called `Characteristic Category` that is assigned to one perspective. To
permit recursive groupings of patterns, `CharacteristicCategory` instances may
contain each other. For example, in the control-flow perspective [6] the patterns are
grouped in six characteristic categories, e.g., structural patterns, cancellation patterns.

Although Figure 3 depicts a reference between the `CharacteristicCategory`
and a `Pattern` by using an association class called `PatternCategoryLink`, a
further refinement of the taxonomy with additional classes is pursued. Thus, a class
`Classifier` organizes a `CharacteristicCategory` with refining keywords
that are commented for clear comprehension. For example, the eSourcing category
called *contractual visibility* [2, 21] is refined by the category keyword *projection*. This

keyword indicates how much process content is projected to an external level where the collaborating counterpart can perceive it. Finally, the class `Assignment` is completing the taxonomy creation by allowing the assignment of a value to a pattern classifier that belongs to a special category. For example, the black-box pattern of the characteristic category with the value *contractual visibility* has the assignment value *none*.

3.3 Pattern-Related Properties

The next package is concerned with properties around class `Pattern`. As Figure 4 shows, a pattern can have one or several `Alias` instances associated, e.g., the control-flow pattern called sequence is also known as *sequential routing* or *serial routing*.
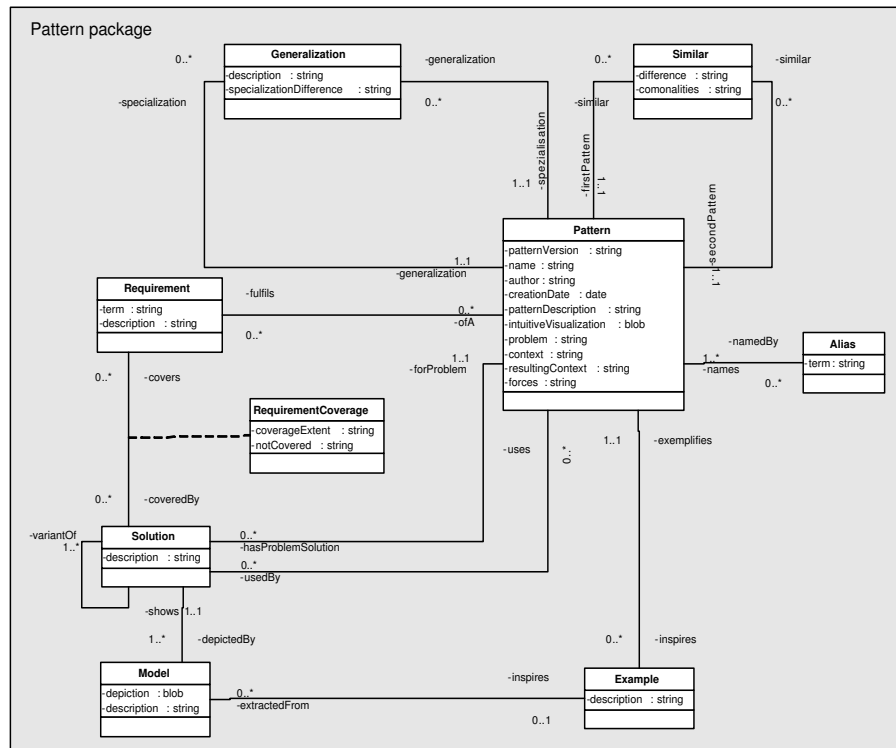
Fig. 4. Detailed class model of the `Pattern` package

Patterns can relate to each other in different ways. Without claiming completeness, the package depicted in Figure 4 includes relationships where patterns are either `Similar` to each other or one pattern is a `Generalization` of another one. For example, it can be argued that the control-flow patterns *parallel split* and *synchronization* are similar as they complement each other where the first pattern creates parallel branches of

execution and the latter pattern is required to join them. On the other hand, the *discriminator* pattern is a generalization of the *n-out-of-m-join* pattern [6]. Both patterns perform a merge of many executing paths and execute the subsequent activity only once. However, while the first pattern performs no synchronization, the latter pattern is more flexible as it is possible to set the number of parallel branches that need to be synchronized.

There are two different ways how a `Solution` may relate to a pattern. Firstly, a pattern tackles a particular problem for which one or many `Solution` instances are applicable. On the other hand a solution is only assigned to one pattern problem. Secondly, it may require the application of several patterns to achieve one greater solution. Therefore a second relationship arc is depicted in Figure 4 where a solution may use several patterns. In this context a solution contains static relationships and dynamic rules describing how to realize a desired outcome. For example, for the eSourcing category contractual visibility [2, 21] the *problem* attribute states the service consumer is interested in using service provision. However, the consumer does not mind how the provision is carried out as long as the specified exchanges are performed correctly. The applied pattern is called *black box* from the *contractual visibility* category and is characterized by a disclosure of business-process interfaces only without revealing the rest of internal process content. A solution that tackles the pattern problem is the application of a three-level framework [16] where it is possible to disclose on an external layer less than the full content of an internally given process on an internal layer.

The pattern and solutions are additionally linked by the `Requirement` class. Thus, a pattern must fulfill some requirements so that a solution becomes applicable. It represents a statement about the demanded pattern function and performance with respect to quantitative and qualitative features. For example, a requirement term can be *system-interoperability support*. A complementary description may mentioned that a solution must pay attention that collaborating parties do not want to disclose all their system details to each other.

The association class `RequirementCoverage` allows textual statements expressing to which extent a solution covers a requirement or not. For example, system-interoperability support is fully covered by a solution that represents a three-layer framework with internal process levels and conceptual process levels for the respective collaborating parties, and one external level for process merging [16]. That way collaborating parties have the chance to hide internal details from each other while only needing to disclose what is necessary to perform process merging on an external level.

Two more classes are depicted in Figure 4, namely the classes `Example` and `Model`. While one example only belongs to one particular pattern, a pattern may inspire several examples. An example is a textual description of a concrete instance in a real-world setting where a pattern is used. Alternatively, it is also possible to give an abstract example that is based on a formal model. An instance of `Model` is a simplified visualization of a reality extract that serves as an example.

While patterns are conceptually formulated, IKWs need to evaluate which patterns are applicable for their own system setup. Thus, the meta model should also cater for the management of technology-support information. In the next subsection a package is presented with classes for capturing such information.

3.4 Capturing Pattern-Support

In Figure 5 a package called `Support` is depicted containing classes for capturing
technology information of patterns. On top of Figure 5 the `Pattern` package is shown
with the subset of contained classes that have relationships to classes from the `Support`
package. Accordingly, a pattern example is visualized by an `Illustration` instance,
a model is expressed in an instance of a `Language` subclass, and a pattern is supported
by an `Artifact`. In the latter case an artifact can be of a complex nature, e.g., a soft-
ware system that supports a standard language.
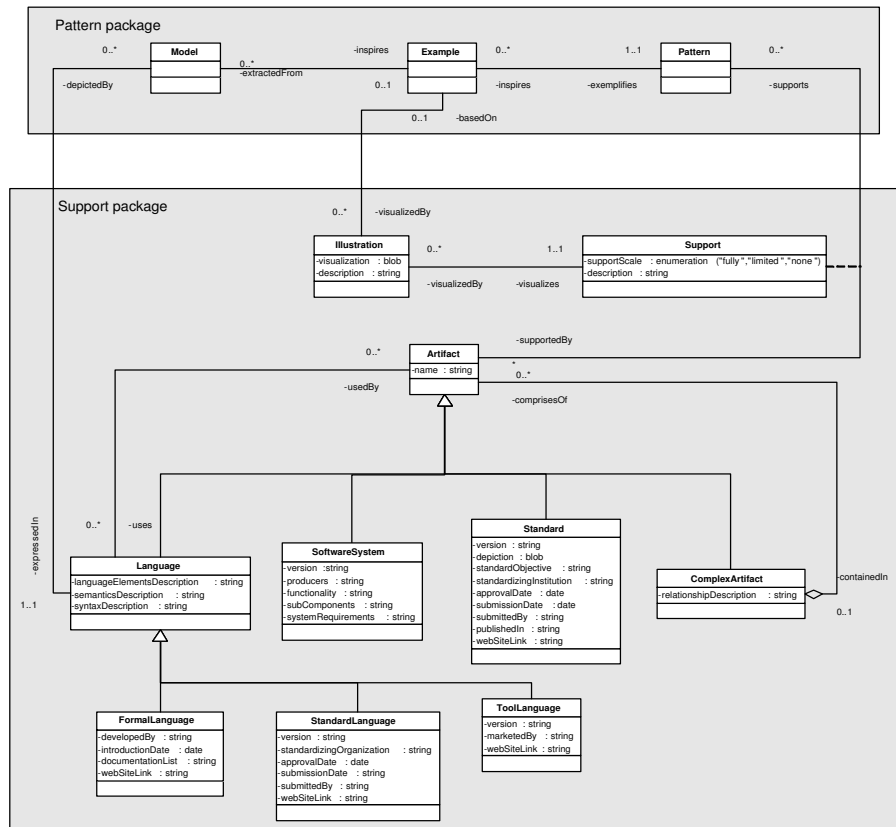
Fig. 5. Detailed class model of the `Support` package

Instances of the class `Illustration` contain a screenshot with textual description
that shows how an artifact supports a pattern, e.g., a modelling element description.
This screenshot is related to an instance of the association class `Support` that links a
pattern instance with a particular technology, indicating to which degree the the pattern
is supported, i.e., either fully, limited, or not at all.

A `Model` can be represented in a language that consists of several elements. One hierarchy level lower, Figure 5 depicts several subclasses. An instance of `Formal Language` is, e.g., the Petri-net markup language PNML [18, 24] that is based on Petri-net theory. Instances of `StandardLanguage` refer to languages that are, e.g., XML based and pursue the objective of supporting the modelling of business processes that are carried out with the help of web service orchestration. Examples of such XML-based standards are BPEL, WSDL, UDDI, and so on. Instances of `ToolLanguage` are related to some software system, e.g., Staffware. Next, an instance of `SoftwareSystem` is an aligned set of programs and application software that perform a specific function directly for the user. Finally, a `Standard` is a prescription or regulation that is fixed by approved institutions. Standards achieve the unification of artifacts that are published in journals. Standards in the ICT domain are necessary for interworking, portability, and reusability. They may be de facto standards for various communities, or officially recognized national or international standards. An example for a standard is the TCP/IP protocol.

In Section 2 it is emphasized that IKWs need support by a pattern repository during creating inter-organizational business process collaboration. The next section explores the features of such an application.

4 An DIBPM Design Application

After explaining a meta model for capturing relevant facts of patterns belonging to differing DIBPM perspectives, the lifecycle of a pattern is presented. Briefly, a pattern must first be reviewed by a committee before it becomes a quality pattern that is available for IKws. The pattern lifecycle is chosen as a starting point for deducting an extension of the pattern meta model. This extension is necessary for capturing additional information that is required for running an online application on top of the pattern meta model. The modules of the application architecture are deducted from the pattern lifecycle.

4.1 A Pattern Lifecycle

In the depiction of Figure 6, an activity diagram shows the lifecycle of a pattern, which starts with an initial pattern proposal that is submitted to the repository by a user who has the appropriate authorization level. This initial version of a pattern needs to go through a process of quality assurance before it may be accessible to IKWs who want to search for quality pattern information. Without a review procedure for patterns, the pattern content of the repository may lack quality.

Thus, a repository user with the authorization of leading a review proposes the pattern for a review process. Repository users with the right skills may volunteer for a review or be explicitly invited by the review leader. Based on a defined review rule, a certain number of review results needs to be submitted for determining whether the pattern is accepted or not. If the review rule is not satisfied, the pattern is rejected and needs to be rewritten as a new proposal. If the review rule is satisfied, the pattern proposal is officially accepted and experiences a status change. Thus, it turns into a quality
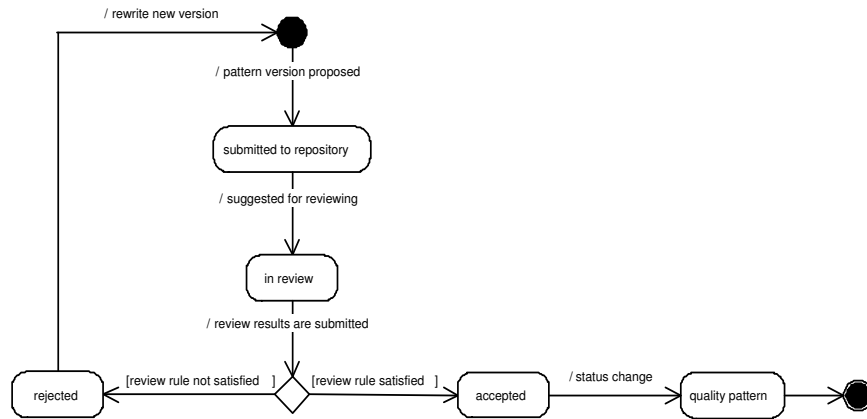
Fig. 6. The lifecycle of a pattern

pattern that is exposed to IKWs for searching. Based on the pattern lifecycle of Figure 6, it is possible to extend the pattern meta model so that relevant data is captured for running an online application on top.

4.2 Extending the Meta Model for User and Review Management

In Figure 7 the `User` package contains relevant classes for handling user and review-related information. On top of Figure 7 the `Pattern` package contains the `Pattern` class, which is the only connection between the two packages.

The central class of the `User` package is called `RepositoryUser` and contains the attribute `volunteer` for indicating whether a user wants to be a reviewer or not. The user of a repository may slip into several `Roles` which have different authorization levels included that influence how a user can interact with the pattern repository. For example, an analyst is only allowed to browse for information but not allowed to edit a pattern.

The model in Figure 7 also depicts the class `Qualification` that is referenced by `RepositoryUser`. The qualifications of a user are relevant for taking part in reviews. Users who do not have required qualifications can not be considered for reviewing patterns. Furthermore, if a user has the appropriate role assigned for leading a review, she can issue invitations for several reviews. Such a repository user also needs to define a `Rule` for a review if a predefined rule doesn't exist already. For example, a rule could state that four reviewers out of five must accept a pattern proposal for becoming a quality pattern while two rejections are enough to totally reject the pattern proposal without waiting for the results of the remaining reviewers.

A `Review` is initiated by a repository user who has the role of a review leader. After assigning a rule to the review, the review leader must find review participants. First, a review leader can look in the pool of users who issued an entry in `VolunteerReview`.
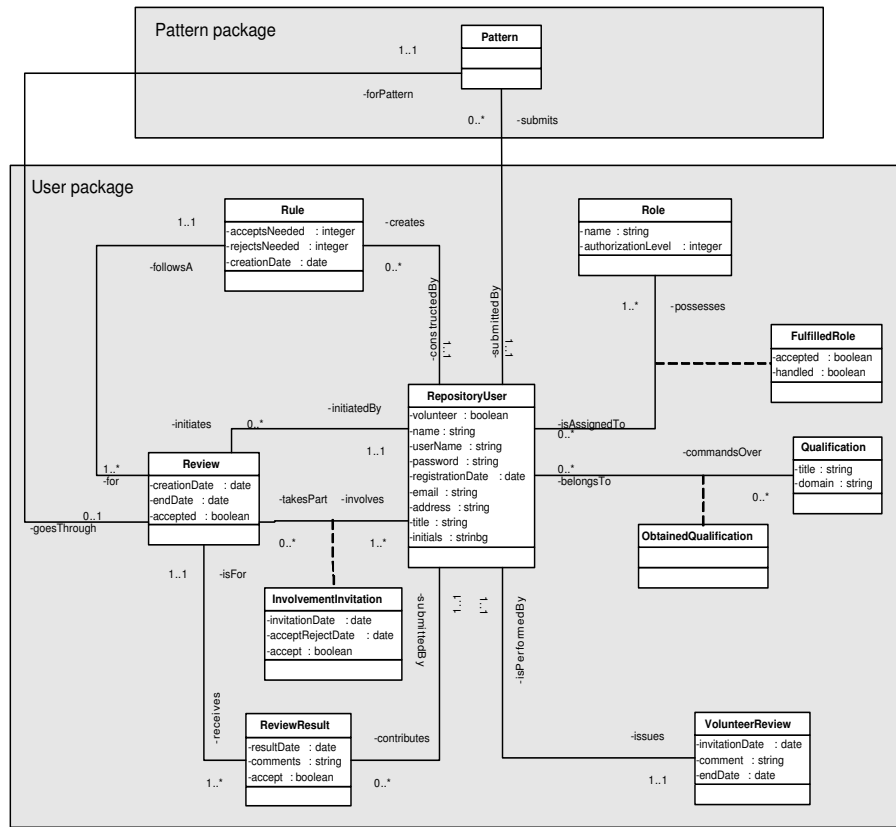
Fig. 7. Detailed class model of the `UserManagement` package

Provided a volunteers possess the appropriate qualifications, a review leader creates an entry in class `InvolvementInvatiation` and therefor establishes a relationship between a repository user and a pattern review. Secondly, if not enough volunteers are available, the review leader searches for rightly qualified repository users and issues a review-involvement invitation that is either accepted or rejected. If the repository rejects the invitation, the entry created by the review leader is removed from class `InvolvementInvatiation`. Finally, once the required amount of review participants is established, entries in class `ReviewResult` are performed that indicate whether a repository user accepts or rejects a pattern proposal.

After deducting an extension for the pattern meta model based on the lifecycle of a pattern, the prerequisite exists for proposing a reference architecture for an online-DIBPM design application. That way IKWs are enabled to interact with the pattern repository.

4.3 An Application Architecture

The pattern lifecycle of Figure 6 shows which types of application users an application architecture must support. Firstly, an *author* is a user who submits a pattern to the repository. A *review leader* forms a review committee for the evaluation of newly submitted patterns. Registered users of the repository who indicate to be volunteers as *reviewers* are invited by the review leader to form a committee. An IKW with the role termed *analyst* is interested in browsing a repository that contains patterns of different perspectives and corresponding information about their artifact support. As indicated in Figure 1, that pattern information helps an analyst to estimate which patterns collaborating business parties support despite their heterogeneous system environments. That way the setup time of inter-organizational business processes is accelerated. Finally, an *administrator* of the pattern repository is required to grant roles to registered users, troubleshoot during pattern reviews, and so on.
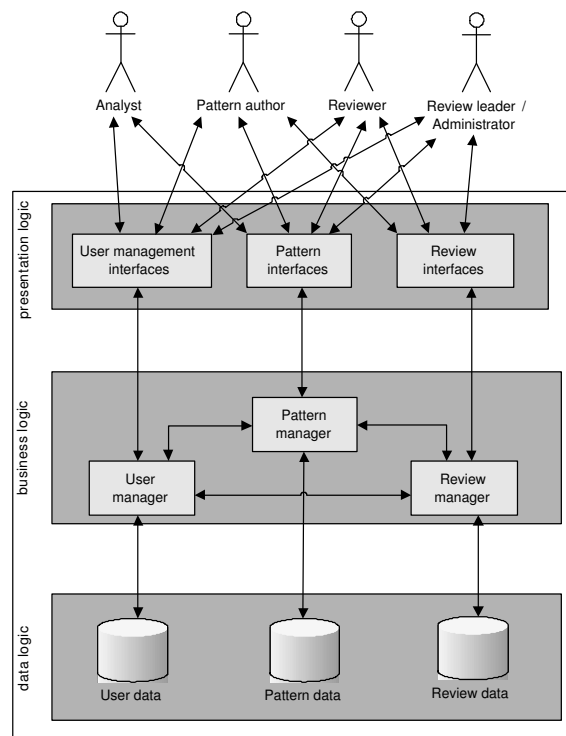

Fig. 8. The application architecture of the pattern repository

The mentioned roles for repository users are input for an architecture on top of the pattern meta model. The described repository user types are depicted in Figure 8 where

bi-directional arrows indicate an exchange with certain modules of the application's web interface. In the interface layer, modules are contained, for user management related interfaces, pattern related interfaces, and review related interfaces.

- The *user management interfaces* offer a repository user to register, claim various qualifications, and request particular roles. As different roles give a repository user different rights, the administrator may need to authorize the roles. Once a repository user is approved, the user management interfaces allow user to login and logout, and modify roles and qualifications.
- The *pattern interfaces* allow users to browse the repository for patterns with a search engine that uses facts from the classes belonging to the taxonomy package (see Figure 3) and the support package (see Figure 5). The generated lists of patterns can be individually selected for exploring their details.
- *Review interfaces* allow a review leader to set up a review committee consisting of reviewers who either volunteer or are appointed based on their qualifications. In the latter case an appointed reviewer may decline through an interface. After the reviewers explore the properties of a pattern, they submit an accept or reject and their feedback for the pattern author. The latter repository user checks the feedback from the reviewers through another interface.

The functionality layer of Figure 8 shows modules that support the web-interface layer, namely the *user manager*, *pattern manager*, and the *review manager*. They process input of the repository users and control the sequence of interfaces that are presented for the signing up and signing in of users, submitting and browsing patterns, performing reviews, and various administration activities. Figure 8 depicts the modules of the functionality layer are referencing each other. For example, to perform a review, the review-manager module uses functionality contained in the user-manager module. As a result competent review teams are organized with the right qualifications. During a review, functionality from the pattern manager allows a reviewer to explore the context a pattern proposal is embedded in, i.e., the taxonomy location, technology support, relationship with other patterns, and so on. Furthermore, a reference between the pattern-manager module and the user-manager module exists for the same reason of employing functionality from each other. For example, if a repository user wants to browse for pattern information, she needs to have the role of an analyst, which must be checked by using functionality from the user manager.

The bottom of Figure 8 depicts the data layer showing databases for *user data*, *pattern data*, and *review data*. These databases are referenced by the corresponding modules of the functionality layer. The figures of Section 3 contain data elements that are in the pattern data. The review and user data is depicted in Figure 7 without claiming completeness.

4.4 A Pattern-Repository Prototype

The reference architecture presented in Subsection 4.3 results from experience based on implementing a pattern-repository prototype. The screenshot of Figure 9 shows a user interface for uploading a visualization of a pattern, e.g., a jpg, gif file, or so on. It

depends on the role a user slips into whether it is possible to upload images. Users who merely have the right to browse the repository for patterns but are not allowed to upload pattern proposals are not offered the user interface below.
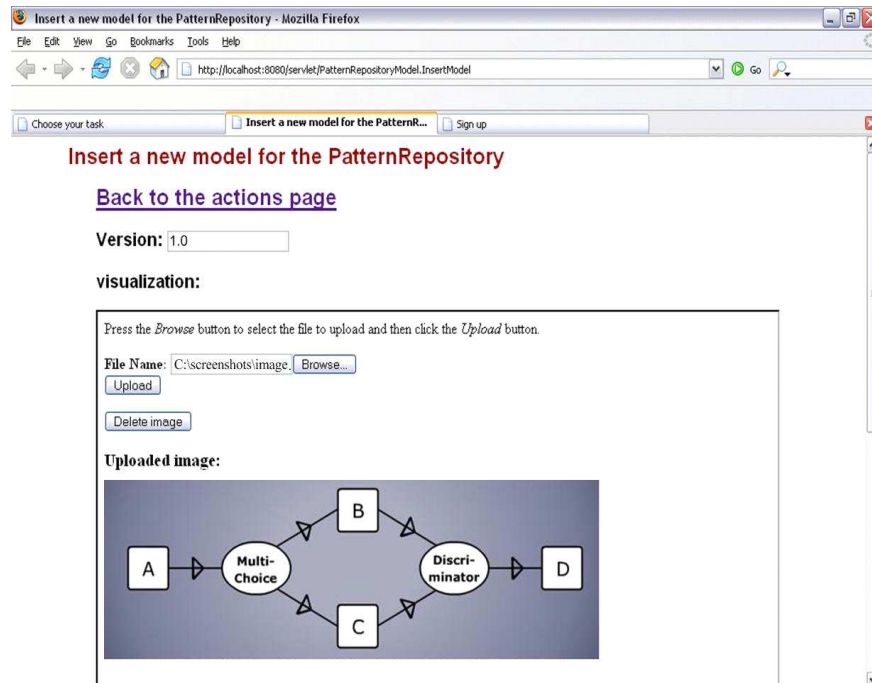

Fig. 9. A screenshot of the pattern repository web interface

The prototype uses ORACLE 10g as a database that is connected to the web via Java Servlet technology. In its current state the prototype implements the full pattern meta model presented in this paper. With respect to the business- and presentation-logic, sufficient functionality is realized for the prototype to serve as an exploratory tool for detecting requirement, of a DIBPM design application.

The following section presents online patterns of different DIBPM perspectives and ongoing research projects where the pattern repository proposes itself as a valuable support.

5 Related Work

Referencing patterns, Gamma et al. [14] first catalogued systematically some 23 design patterns that describe the smallest recurring interactions in object-oriented systems. Those patterns are formulated in a uniform specification template and grouped

into categories. For the domain of intra-organizational business process collaboration patterns were discovered in various perspectives.

In the area of control flow, a set of patterns was generated [6–8] by investigating several intra-organizational workflow systems for commonalities. The resulting patterns are grouped into different categories. Basic patterns contain a sequence, basic splits and joins, and an exclusive split of parallel branches and their simple merge. Further patterns are grouped into the categories advanced branching and synchronization, structural patterns, patterns involving multiple instances, state-based patterns, and cancellation patterns. The resulting pattern catalog is for the evaluation [5, 25] of WSCLs.

Following a similar approach as in the control-flow perspective, data-flow patterns [22] are grouped into various characteristics categories. One category is focuses on different visibility levels of data elements by various components of a workflow system. The category called data interaction focusses on the way in which data is communicated between active elements within a workflow. Next, data-transfer patterns focus on the way data elements are transferred between workflow components and additionally describe mechanisms for passing data elements across the interfaces of workflow components. Patterns for data-based routing deal with the way data elements can influence the control-flow perspective.

Patterns for the resource perspective [23] are aligned to a the lifecycle of a work item. A work item is created and either offered to a single or multiple resources. Alternatively a work item can be allocated to a single resource before it is started. Once a work item is started it can be temporarily suspended by a system or it may fail. Eventually a work item completes. The transitions between those life-cycle stages of a work item either involve a workflow system or a resource. Characteristic categories for the resource perspective are deducted from those life-cycle transitions and group specified patterns.

So-called service interaction patterns [9] are specified for the coordination of collaborating processes that are distributed in different, combined web services. Again, the patterns are categorized according to several dimensions. Based on the number of parties involved, an exchange between services is either bilateral or multilateral. The interaction between services is either of the nature single or multi transmission. Finally, if the bilateral interaction between services is of the nature two ways, a round-trip interaction means the receiver of a response must be equal to the sender. Alternatively a routed interaction takes place.

Many other e-business related patterns are textually available online [3] for the perspectives business-, integration-, composite-, custom design-, application- and runtime patterns. The business perspective highlights the most commonly observed interactions between users, businesses, and data. Integration patterns connect business patterns for creating composite patterns. Patterns of the composite perspective that combine business and integration patterns are only documented when they often occur in reality. The perspective custom design is similar to composite patterns. Finally, patterns from the application perspective focus on the partitioning of the application logic and data while patterns from the runtime perspective use nodes to group functional requirements that are interconnected to solve a business problem.

The INTEROP [4] network of excellence comprises a task group for methods, requirements and method engineering for interoperability. It is the objective of that task group to develop and validate ways of providing a knowledge repository of interoperable method engineering services. The pattern repository is to be integrated during the implementation of the method-chunk repository.

6   Conclusion

This paper proposes that intra and inter-organizational knowledge workers should employ patterns for dynamic inter-organizational business process management. Using patterns promises the speedy evaluation and integration of intra-organizational business processes across the domains of collaborating parties. Since many patterns are specified in different perspectives for DIBPM, the need for a knowledge system in the form of a pattern repository arises to support IKWs. Thus, this paper describes a meta model for uniformly storing pattern specifications, orders them in a taxonomy, and caters for capturing information about technology support of specific patterns. Furthermore, an extension of the pattern meta model allows to store data about repository users and pattern reviews. An architecture for an online application is presented that builds on top of the pattern meta model.

With respect to ongoing research projects, the pattern repository is part of the proof-of-concept architecture for the EU project called CrossWork [1]. It is the objective in CrossWorks to develop automated mechanisms for allowing dynamic workflow formation and enactment, enabling hard collaboration and strong synergies between different organizations. Software agents in CrossWork employ a knowledge base for reasoning about automated workflow formation. Thus, by extending the pattern repository proposed in this paper with a formal second tier, agents can use facts about patterns for workflow formation. Furthermore, the pattern repository proposes itself as as an integral part of a method-chunk repository that is built in the framework of the task group for methods, requirements and method engineering for interoperability of the INTEROP network of excellence.

Scope for future research exists for the repository prototype where the development of sophisticated graphical user interfaces constitutes a problem of considerable complexity. Furthermore, the prototype must comprise a powerful search engine that permits an inter-organizational knowledge worker to intuitively find suitable patterns with diverse combinations of data.

References

1. CrossWork: Cross-Organisational Workflow Formation and Enactment. http://www.crosswork.info/.
2. eSourcing: electronic Sourcing for business to business. http://is.tm.tue.nl/research/eSourcing.
3. IBM patterns for e-business. http://www-128.ibm.com/developerworks/patterns/.
4. INTEROP: Interoperability Research for Networked Enterprises Applications and Software. http://interop-noe.org/.

5. W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede, and P. Wohed. Pattern-Based Analysis of BPML (and WSCI). *QUT Technical report*, (FIT-TR-2002-05):487–531, 2002.

6. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns Home Page. http://www.workflowpatterns.com.

7. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Advanced Workflow Patterns. In O. Etzion and P. Scheuermann, editors, *7th International Conference on Cooperative Information Systems (CoopIS 2000)*, volume 1901 of *Lecture Notes in Computer Science*, pages 18–29. Springer-Verlag, Berlin, 2000.

8. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(3):5–51, 2003.

9. A. Barros, M. Dumas, and A.H.M. ter Hofstede. Service interaction patterns. In W.M. P. van der Aalst and F. Curbera B. Benatallah, F. Casati, editors, *Business Process Management: 3rd International Conference, BPM 2005*, number 3649 in Lecture Notes in Computer Science, pages 302–318, Nancy, France, 2005. Springer Verlag, Berlin.

10. BPML.org. *Business Process Modeling Language (BPML) version 1.0*. Accessed August 2003 from www.bpmi.org, 2003.

11. F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte, and S. Weerawarana. *Business Process Execution Language for Web-Services*. http://www-106.ibm.com/developerworks/library/ws-bpel/, 2003.

12. J.L.G. Dietz. The deep structure of business processes. *Communications of the ACM*, 49(5):58–64, 2006.

13. F. Flores, M. Graves, B. Hartfield, and T. Winograd. Computer systems and the design of organizational interaction. *ACM Transactions on Information Systems (TOIS)*, 6(2):153–172, 1988.

14. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Professional Computing Series. Addison Wesley, Reading, MA, USA, 1995.

15. P. Grefen. Service-Oriented Support for Dynamic Interorganizational Business Process Management. to appear, 2006.

16. P. Grefen, H. Ludwig, and S. Angelov. A Three-Level Framework for Process and Data Management of Complex E-Services. *International Journal of Cooperative Information Systems*, 12(4):487–531, 2003.

17. S. Kimbrough and S. Moore. On automated message processing in electronic commerce and work support systems: speech act theory and expressive felicity. *ACM Transactions on Information Systems (TOIS)*, 15(4):321–367, 1988.

18. E. Kindler and M. Weber et al. *Petri Net Markup Language (PNML) Home Page*. http://www.informatik.hu-berlin.de/top/pnml/, 2003.

19. A. Norta and P. Grefen. A Framework for Specifying Sourcing Collaborations. In Jan Ljungberg and Bo Dahlbom, editors, 14th *European Conference on Information Systems: Grand Challenges*, pages CD–ROM, Gothenburg, Sweden, 2006. IT-University Gothenburg.

20. A. Norta and P. Grefen. Developing a Reference Architecture for Inter-Organizational Business Collaboration Setup Systems. BETA Working Paper Series, WP 170, Eindhoven University of Technology, Eindhoven, 2006.

21. A. Norta and P. Grefen. Discovering Patterns for Inter-Organizational Business Collaboration in a Top-Down Way. BETA Working Paper Series, WP 163, Eindhoven University of Technology, Eindhoven, 2006.

22. N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Workflow Data Patterns. *QUT Technical report*, (FIT-TR-2004-01), 2004.

23. N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Workflow Resource Patterns. *BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven*, 2004.

24. M. Weber and E. Kindler. The petri net markup language. In H. Ehrig, W. Reisig, G. Rozenberg, and H. Weber, editors, *Petri Net Technology for Communication-Based Systems Advances in Petri Nets*, number 2472 in Lecture Notes in Computer Science, page 455 p. Springer Verlag, Berlin, 2003.

25. P. Wohed, W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede. Analysis of web services composition languages: The case of bpel4ws. In I.Y. Song, S.W. Liddle, T.W. Ling, and P. Scheuermann, editors, *22nd International Conference on Conceptual Modeling (ER 2003)*, number 2813 in Lecture Notes in Computer Science, pages 200–215, Chicago, Illinois, 2003. Springer Verlag, Berlin.