2002

Specifying and Analyzing Static and Dynamic Patterns of Administrative Processes

Renate Motschnig-Pitrik
University of Vienna, motschnig@ifs.univie.ac.at

Philipp Randa
University of Vienna, randa@ifs.univie.ac.at

Guenther Vinek
University of Vienna, vinek@ifs.univie.ac.at

SPECIFYING AND ANALYSING STATIC AND DYNAMIC PATTERNS OF ADMINISTRATIVE PROCESSES

Renate Motschnig-Pitrik, Philipp Randa, Günther Vinek

University of Vienna
Department of Computer Science and Business Informatics
Rathausstr. 19/9, A-1010 Vienna, Austria
Tel. (+43)(+1) 4277 38415, fax: (+42)(+1) 4277 38428
Email: motschnig@ifs.univie.ac.at, randa@ifs.univie.ac.at, vinek@ifs.univie.ac.at

ABSTRACT

In the last decade, patterns have proved their effectiveness for the reuse of software development artifacts from early as well as late phases of the process. This paper proposes analysis patterns based on accurate and thorough observations of administrative processes such as organizing meetings, making group decisions, hiring of personnel, etc. The patterns proposed in this work are characterized by coming equipped with a basic workflow, captured in the form of an UML activity diagram, aside of providing an object structure modeled as UML static structure diagram. Besides serving as analysis models for subsequent software development, the primary intent behind administrative patterns is to visualize and to standardize frequently reoccurring processes and thus to support quality assurance and - control. Furthermore, administrative patterns lay the basis for various ways of evaluating the underlying processes in order to improve them.

1   INTRODUCTION

Starting from architecture (Alexander et al. 1977), patterns rapidly spread from capturing object structures occurring at the design phase of software development (Gamma95) to higher level structures found during analysis (Fowler 1997). More recently, patterns have been proposed to capture aspects like usability (Perzel and Kane 1999) and navigational structures of web applications (Rossi98). While the degree to which patterns describe aspects of some particular application differs significantly, the primary intent behind patterns, namely to provide useful molecular abstractions of some reoccurring phenomenon for the sake of reuse stays constant across individual approaches.

Our approach to mining patterns has been motivated by the observation that part of our work at the university that includes administration could be made more efficient if supported by well documented procedures and potentially also a software architecture and tools supporting these procedures. Concretely, if we were about to plan a meeting, we should be given easy access to information regarding the availability of appropriate rooms and a view of the calendars of persons. At this point the reader will have noticed that although our most prominent viewpoint on administration is the university, the processes we aim to capture in the form of patterns have an impact that reaches far beyond the university context. It should become clear soon that the patterns described will be found useful for any area where people work together in some disciplined way.

In particular, we perceive the current situation regarding administrative processes to be characterized by the observation that thousands (of thousands) of people need to follow some procedure that they are knowledgeable about only informally or very coarsely, details being spread across several places such as books, booklets, regulations, standards, or even the heads of administrative personnel. This distribution and hard access to knowledge about processes leads to the effect that similar processes are described in different forms at different places. As a result, it is hard to

862

check the consistency between similar processes making it difficult to compare, support, evaluate, and to improve them. We are convinced that capturing administrative patterns and visualizing and documenting them has several far-reaching benefits. A major advantage, particularly in large organizations, is the reuse of knowledge captured in the patterns to be specialized in the descriptions of concrete procedures. The latter can then be modeled visually and published, say, in web-pages, such that they are easily accessible on demand. Further, patterns can serve as inputs for building alerting services as well as for tools supporting workflow enactment. Classically, they can be seen as sources for cooperative work as well as consistent and succinct artifacts driving the documentation of processes as demanded by ISO-9000-3 standards in the context of quality assurance and -control.

Traditionally (Gamma 1995), pattern specifications consist of the following components:

- Name: should be well chosen and descriptive regarding the problem area
- Problem: describes the basic problem and the context for the application of the pattern
- Solution: describes the objects and/or classes that constitute the solution along with their interdependencies; provides a template that can be specialized on demand.

Further, a pattern shall be applicable to at least three examples with different backgrounds.

Our approach to patterns complements pattern specification by another dimension, namely the workflow being abstracted into the pattern. Compared with approaches described in the literature, our proposal most closely matches Ambler's Process patterns (Ambler 1998) in so far, as procedural aspects are the guiding principles in discovering and identifying a pattern. Different to Process Patterns, whose subject of discourse is the software development process, our patterns concern the broad field of administrative workflows that occur at any occasion where people work together or, even more generally, organize themselves into a group to fulfill some particular purpose. Our approach most significantly differs from previous work by its basic procedural orientation. The patterns proposed in this work all abstract from some specific administrative workflow, such as organizing a meeting or making a group decision. Thus, by their mere nature, they are procedural. This procedural orientation is reflected by providing a workflow specification as part of each administrative pattern. Regarding the formalism for the specification of the respective workflow we found UML activity diagrams most appropriate (Rumbaugh et al. 1999). This is because they allow one to model activities, conditions under which they are performed and, additionally, also the actors that perform the activities. The latter are specified in the form of swim lanes. Furthermore, the possibility to describe abstract activities that are amenable to specialization makes activity diagrams favorite candidates for workflow modeling.

At a first quick glance our approach may seem similar to the one described in (Eriksson and Penker 2000). The process patterns presented there are high-level patterns about the modeling of processes in general. However, our procedural patterns rather use these process patterns as guidelines for their specification and hence can be seen as realizations of Eriksson and Penker's general process patterns. Another difference between our approach to patterns when compared with their traditional use is their secondary purpose. At this point recall that the primary purpose is that of reuse that appears equally in our proposal. While traditional patterns, in a broad sense, support software development, administrative patterns follow this target only in as a secondary purpose. In the first place, they are targeted at visualizing and documenting administrative processes in order to make them explicit, transparent, and easy to follow and thus to lay a foundation for a successive comparison and evaluation of process variations. This subsumes goals such as to visualize the processes, to clarify them, to facilitate their reuse, to communicate them to new employees, and, importantly, to support them in effective ways. Note that, as a positive side-effect, this approach supports quality assurance as well as -control.

Last but not least we try to develop a validation concept to help to evaluate the quality of a certain process. We propose criteria to be considered in order to determine, at the design stage, whether a process meets basic quality requirements. In general, our approach shares important goals with the

area of business process reengineering (BPR). These are, in particular, the modeling of business processes in order to evaluate and to improve them. Our approach, however, can be seen as complementary to BPR in at least two ways: First, we propose patterns of business processes as generalizations of processes classes with individual processes being instances of these classes. Second, we search for evaluation criteria other than algorithmic procedures and simulation mechanisms almost routinely applied in the area of BPR.

This paper presents some administrative patterns in more detail, describes others in brief and discusses their use. In the next section we present and motivate the descriptive framework that will be used in the subsequent part of the part in order to present and to discuss administrative patterns. Section three then specifies selected patterns and their relationships and illustrates application scenarios involving administrative patterns. Section four briefly introduces criteria for evaluating basic quality characteristics. The final Section discusses our approach and points to further work.

2 INTEGRATING PATTERNS WITH ADMINISTRATIVE WORKFLOWS

As already indicated above, the peculiarity of our approach is to combine procedural aspects with structural ones within each pattern. This is achieved by providing two views, a dynamic and a structural one. While the first is represented as UML activity diagram(s), the latter is specified in the form of a UML static structure diagram, respectively (Eriksson and Penker 2000), (Rumbaugh et al. 1999). This way of capturing both dynamic and static regularities in administrative workflows has several benefits that are briefly listed below:

- Like with all kinds of patterns the most evident benefit of our approach is the reusability of administrative patterns. Different to several other approaches we aim to reuse analysis-level knowledge.
- The patterns are quite easy to understand, because most of the models represent basic and frequently occurring problems/procedures.
- They can be used to reflect the reason why certain process sequences are similar and thus represent instances of only one basic process (such as buying a ticket for the cinema and enrolling a student to a course).
- They represent a good basis for understanding particular tasks. In general, only small adaptations to the basic patterns suffice, to reflect real-world processes.
- They are more complete than other patterns, in the sense of covering both structural and procedural knowledge.
- They serve modeling, specification and evaluation purposes prior to driving software development.
- By combining different patterns it is possible to create a framework, which represents more than the sum of the individual patterns.
- They help to reduce errors, for example in the form of omissions, in running their referent real-world processes.
- They represent an efficient and tested solution base for a specific but very broad domain.

In order to prepare the reader for the discussion of individual patterns, let us walk through the descriptive framework of Randa (Randa 2001) designed to specify the individual patterns. We consider it as important to provide the reader not only with the diagrams, but in addition with information about precondition, goal, decision, and limitations for every specific pattern.

Pattern Name
The name of a pattern is quite important, because it shall indicate the intent of the pattern and shall be used in discussions about the pattern. That's why the name has to be short and expressive (and is often hard to find).

864

Intent

Provides the reader with information about problem area(s) in which the pattern may be used and what it is supposed to achieve.

Motivation

By using a descriptive scenario for the specified problem area and by showing how the pattern contributes to the problem's solution, the subsequent pattern specification should be easier to understand.

Solution

Describes the pattern's content and the associated solution structure.

> **Precondition**
>
> Specifies the conditions, problems, or triggers that indicate the appropriateness of employing this pattern. The preconditions are described in the form of a note-symbol at the head of each activity diagram.
>
> **Goal**
>
> Shows which goals can be reached by using the particular pattern. Like the precondition it is described as a note, at the head of each activity diagram.
>
> **Static and Dynamic Structure**
>
> This section represents the core of the pattern description. It shows the dynamic and structural models, employing UML's activity- and class diagrams, respectively. The activity diagrams use swim lanes to capture those actors (better actor roles) that perform individual activities. The latter are represented in the activity diagram along with their temporal order. The structural and static aspects, in other words the data structure, are captured with a static structure diagram often referred to as a class diagram. This diagram encompasses the methods required to handle and to administer the respective data. Besides the graphs, textual descriptions are available to capture and to illustrate the working of the pattern.

Examples

Shows examples for the usage of the pattern in the real world. It should help the reader to better understand the pattern and to prove its applicability. The examples chosen are (or should be) derived from different backgrounds.

Related Patterns

Lists other patterns that are related and/or can be concatenated with this pattern. In the case that the actual pattern uses sub-patterns they are also listed in this section.

3   ADMINISTRATIVE PATTERNS AND THEIR RELATIONSHIPS

Our initial endeavors so far have led us to suggest a small catalogue of patterns. Although all patterns have been discovered by looking for similarities in workflows in the context of the administrative environment of the University of Vienna, it soon became clear that their usability reached far beyond the realm of their origin. Also the development of these patterns took a different course than traditionally tends to be the case. Due to the fact that our main interest has been in the procedural flows, the dynamic pattern was first created and only then was the structural part derived. The derivation proceeded by investigating the individual activities input- and output-object needs. These further gave rise to the identification of methods an application would typically need. An essential strategy and desire in the development of the patterns has been to make them atomic and encapsulated in order to allow one to concatenate different patterns in developing solutions for larger problems, without having to create/develop new patterns from scratch. The relationship between a concatenated pattern, i.e. a template for a process block, and its atomic constituents or particles is a particular category of a part-whole relationship (Motschnig-Pitrik 1999).

So far, we developed the following patterns that fall into one of two categories:

1. Atomic patterns:

   Identify date/location, Publishing, Validate, Voting

865

2. Concatenated patterns[1]:
   Make an appointment (+1), Register (for a limited group) (+1), Survey (+1), Meeting (+3)

3.1 The "Register" Pattern

Since space does not permit to describe all patterns in detail, we decided to select, in the first place, only one pattern, "register", to be explained and described in detail along with the sub-pattern "validate", and just to list the remaining patterns, giving only brief explanations. The "register" pattern is particularly illustrative for its concatenation with the "validate" pattern, which, in this context, represents a sub-pattern (to be specified in Section 3.3). There is also a wide range of examples, where this pattern can be applied.

Pattern Name: Register (for a limited group)

Intent

Can be used in all kind of situations, where a person wants to join a group of people, for which he/she has to fulfill particular conditions. Only if these conditions are met, the person can join the group and will get some kind of proof of registration to verify, that he/she is a member of the particular group for a given period of time.

Motivation

Groups tend to have rules or conditions to be met by their members. There is a wide (nearly unlimited) range of groups in all areas of our world. Some examples of the registration procedure for different groups are: admission to study at a university, buying a ticket (flight, cinema, etc.), issuing a new passport, hiring of an employee, etc.

Solution

    Precondition

    A person wants to join a particular group of people, knows all postulated criteria and tries to show that he/she meets these criteria.

    Goal

    Persons are allowed to join a group only, if they meet all required conditions and if there is still at least one place vacant in the group.

    Static and Dynamic Structure

    Activity diagram

    There are only two actors in this pattern: the customer and the tester (most often a clerk)

Order of events:

A customer wants to join a particular group and is informed about the conditions of membership. He/she presents all required documents or whatever has to be shown to meet the membership conditions. The clerk, a special actor who is authorized and has the ability to check all postulated conditions, validates the data received, following the "validate" sub-pattern, as shown in Figures 1 and 3. The validation can result in a positive or a negative outcome. If the result is:

- negative, the clerk informs the customer about the unfavorable outcome. This can be done electronically, in a written form, or orally.
- positive, the clerk registers the customer and provides him/her with a proof of registration, allowing the customer to verify his/her membership for a given period.

Note that the "validate" pattern is not only useful in this context, but can be used on many other occasions.

---

[1] The number in the bracket indicates, how many sub-patterns are used.
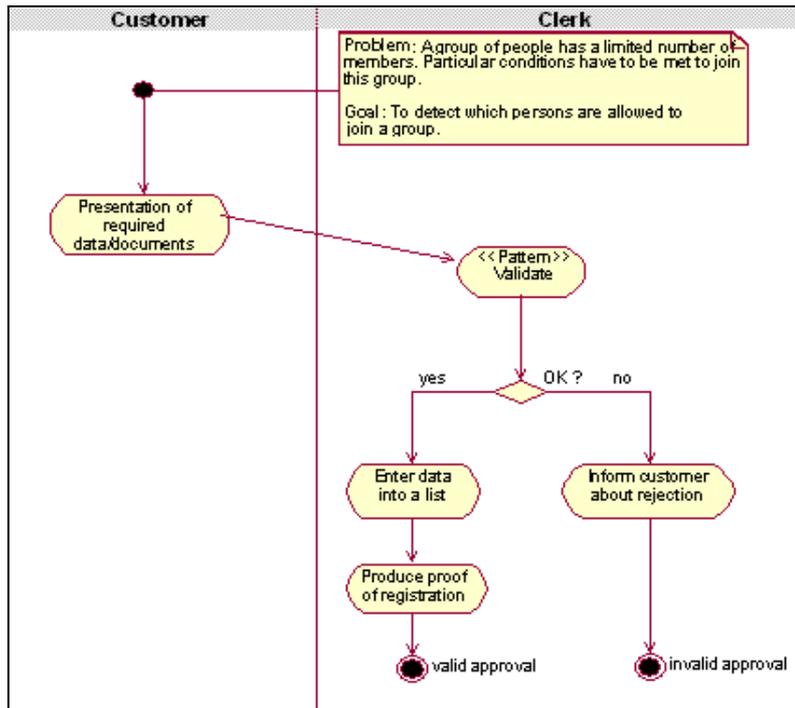
866

Figure 1: UML Activity diagram as part of the "Register" pattern

Class diagram
From the activities described above, the following data structure is derived, whereby classes from the "validate" sub-pattern (see Figure 4) have already been integrated into the pure "register" pattern.
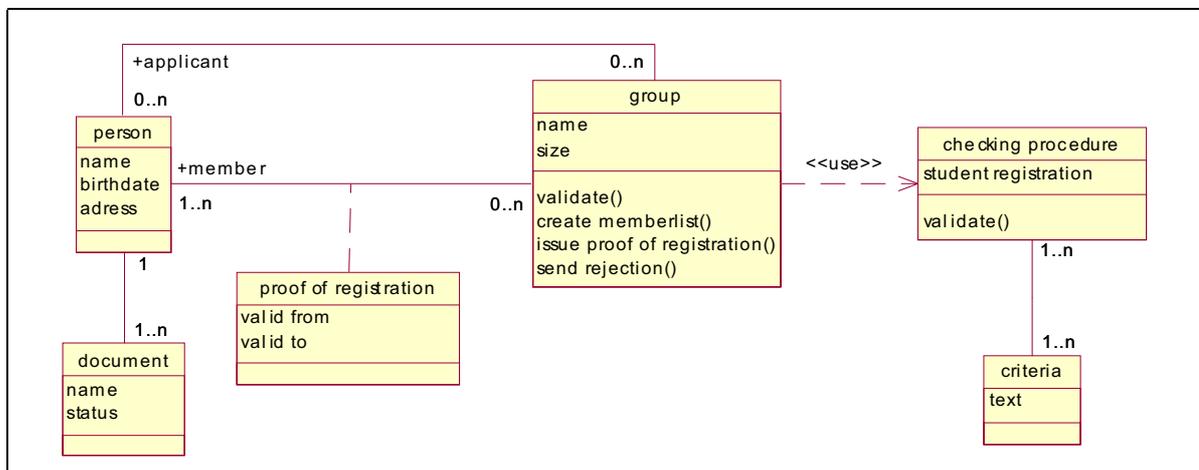

Figure 2: UML static structure diagram as part of the "Register" pattern

This diagram shows, that zero or more persons may apply for membership in a group. Each applicant owns one or more documents, whereby the term document is used in a very generic sense in this pattern. The group uses a particular checking procedure (from the "validate" pattern), which is associated with one or more criteria. The "validate()" method checks, if all documents meet the postulated criteria. In the positive case, an association is built between the person and the group. The resulting "proof of registration", generated by the method "issue proof of registration()", is best described as an association class attached to the person - group association. Its two attributes specify

867

the date/time of the beginning and the end of a membership. Furthermore, the group has a method to create a list of all members and to send a rejection, if the "validate()" method returns a negative result.

Examples

Examples for the use of this pattern are a student's admission to study at a university (explained in detail in Section 3.3), a student's enrolment into a course, the acquisition of a flight ticket, the registration of a user to some web-application, etc.

Related Patterns: Voting, Validate

3.2   The "Validate" Pattern

Pattern Name: Validate

Intent

Any validation process tends to involve criteria that have to be checked, in order to decide whether they are met. The "Validate" pattern can be used for all kinds of validations, based on arbitrary criteria.

Motivation

In many areas and on many occasions it seems necessary to establish whether something matches prescribed criteria and therefore passes the test or otherwise fails it. This check can be motivated by considering different reasons, e.g. laws, regulations of a society, etc. The validation criteria can change over time, but if the validation subject matches the new criteria, it will still pass the check. An example for changing criteria is the enactment of a new law. To perform a validation it is essential that the completeness of all necessary data, documents, etc. be checked in the first place.

Solution

Precondition

Any subject matter like an object, a value, or some abstract concept may be subject to examination. All necessary documents and/or data have to be passed to the person who performs the validation.

Goal

It is assessed, whether specific criteria are fulfilled. The result can be used to determine, whether a given process can proceed or must be aborted.

Static and Dynamic Structure

Activity diagram

Only one actor is needed, that is the checker, who performs the validation. The checker is a special kind of person, who tends to have different qualifications and competences that have considerable consequences on the validation procedure.
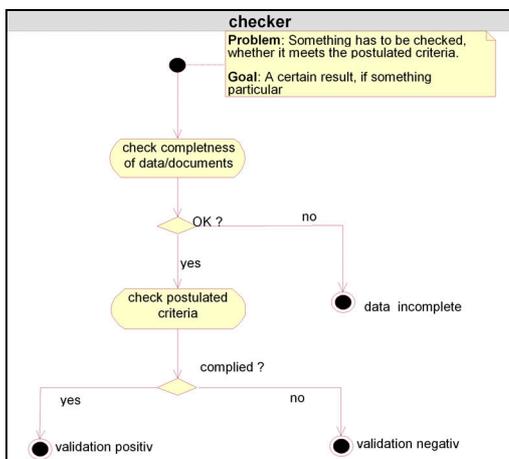

Figure 3: Activities in the "validate" pattern

Order of events:

At first the checker verifies the completeness of the documents and information received. In the case that required "input" items are missing, the checker has to postpone the validation task, until the required items are complete. In the next step it is checked whether all criteria can be met. In the positive case, the validation process is completed successfully. In any other case the validation produces a negative result.
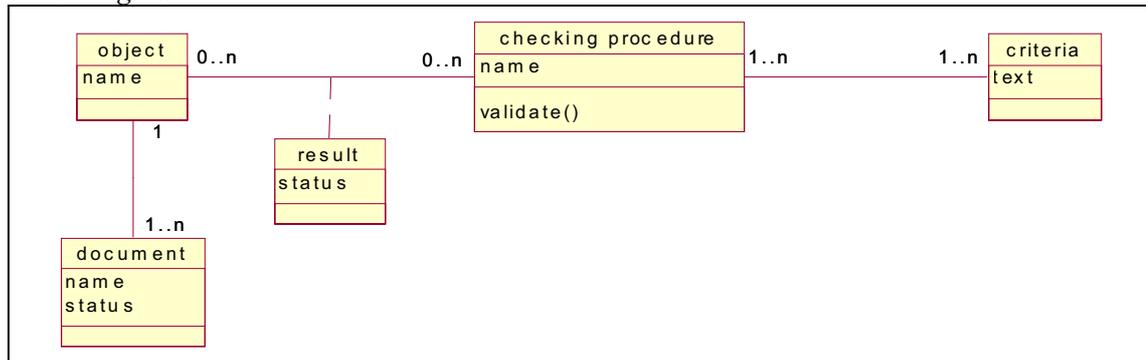
Class diagram

Figure 4: Class diagram of the "validate" pattern

Figure 4 shows a class diagram for the "validate" pattern. Therein the checking procedure is associated with zero or more objects that can represent persons, business cases, or any other phenomena to be checked. The object is associated with one or more documents being required for the validate method. Like already mentioned in the "register" pattern (Section 3.1), the term document is used in a very generic sense. The "status" attribute in the document class can be used to indicate, whether a specific document has already been seen or not (by the checker). The checking procedure is also associated with one or more criteria, which are used by the "validate()" method to check the documents and their included information. The result of the "validate()" method will be saved in the "status" attribute of the association class "result".

Examples

As a first example considers, the validation procedure targeted at determining, whether an applicant can be accepted as a new student at the university. Another example is the check, whether a person is allowed to drive a car, or whether the meat in the supermarket is licensed and fresh enough to be sold.

Related Patterns: Register, survey

4 CRITERIA FOR IMPROVING BUSINESS PROCESS DESIGN

In order to compare alternative business processes and to assess their quality, numerous factors and number systems have been developed and implemented as components of business process design tools (Aichele 1997), (Junginger 1998). Typically, to evaluate a process, individual activities are associated with data on time, cost, and quality and a goal function is determined to drive the simulation process. Thus, it is possible to judge, e.g. after a re-design of a business process, whether it improved with respect of the time, cost and/or quality perspective.

Our research hypothesis, however, says that there exist other factors that determine whether a process can be seen as successful. Some of these factors, which we derived from observation, experience, and also responsibilities for the outcomes of administrative processes, are given below. In particular, our experience with administrative processes showed that there exist a number of criteria that are applicable already in the design phase of business processes. Interestingly, some factors tend

869

to apply for all process classes that follow the same pattern. Thus, we suggest that the following guidelines accompany our design patterns for business processes:

1.) Business processes should be designed in such a way that major factors causing their premature termination should be determined as soon as possible. This is because the more effort has been spent in the process the more costly this process becomes both for the customer and the service provider. As an example, consider the validation process of documents. If one checks each document in detail and only after having inspected several documents realizes, that a required document is missing, time has been wasted.

2.) Preferably, structures well known to a society should be reused, in order to allow one to follow known metaphors. Following this guideline makes processes intuitive and easy to handle even without special knowledge about the particular process.

3.) Possible results of activities should be considered and stored in order to eliminate redundant process sequences. E.g. in the business process "validation of documents" it should be maintained which documents already have been checked, so that another (later) business processes, that likewise requires documents to be submitted, does not check the same documents again.

4.) For business processes with variable load, a concept for peak loads should be developed. For example, an alternative process for peak loads could be designed.

5.) Each business process should be designed for robustness and as reliability. In particular, critical points in a sequence should be determined and workarounds be arranged for the case of failure of an actor. This can be achieved, amongst others, by parallelism of activities.

6.) Business processes should be secured against a manipulation of
    a.   the sequence of activities and
    b.   data
from internal and external. This can be achieved e.g. by allocating authorizations on different actors (four-eyes-principle) or by task-based authorization.

Aside of these guidelines, we aim to find out general criteria for improving the quality of business processes from the customers' point of view. In this respect, we suggest that the temporal interruption of business processes should be minimized. Clearly, the more often a customer has to show up the worse it is for all parties. As another sample consider the guideline to make business processes, whenever possible, transparent to all parties involved, such that the competence and responsibility of actors for the individual activities can be determined. This will result in more effective communication regarding all aspects of the particular process.

5   CONCLUSIONS AND FURTHER WORK

We have identified and developed patterns of administrative workflows that are broadly applicable. Our patterns capture analysis-level knowledge and differ from the traditional approach to patterns in at least two significant ways. Firstly, being developed from observing workflows, they combine dynamic, procedural knowledge, captured in UML activity diagrams with structural knowledge, modeled in UML class diagrams. Secondly, their primary purpose is to visualize workflows and associated artifacts in order to understand, compare and to evaluate them prior to the provision of automated support. Further work will concern the analysis and evaluation of administrative processes according to various criteria as well as automated support for the specialization and adaptation of patterns along the lines of research reported in (Wohed 2000). We also intend to investigate the

870

meaning and role of process patterns, process classes, as well as concrete instances for organization development. In this context we aim to find out what e.g. it means for an organization to implement few versus many process classes following one process pattern. Finally, with this research we also hope to throw some light on the administrative processes of big organizations in order to make the processes more transparent and understandable and thus to facilitate their effective tracking and enactment.

REFERENCES

Aichele C. (1997). Kennzahlenbasierte Geschäftsprozessanalyse. Gabler, Wiesbaden.

Alexander, C., Ishikawa, S., Silverstein, M. (1977). A pattern language. Oxford Univ. Press, New York.

Ambler, S., W. (1998). An Introduction to Process Patterns. An AmbySoft Inc. White Paper. http://www.ambysoft.com/processPatterns.pdf (Apr. 2001).

Eriksson, H-E., Penker. M. (2000). Business Modeling with UML – Business Patterns at Work. Wiley Computer Publishing, New York.

Fowler, M. (1997). Analysis patterns. Addison-Wesley, Menlo Park, California, USA.

Gamma, E. (1995). Design patterns. Addison-Wesley, Reading, Mass., USA.

Motschnig-Pitrik, R., Kaasboll, J. (1999). Part-Whole Relationship Categories and Their Application in Object-Oriented Analysis. *IEEE Transactions on Knowledge and Data Engineering*, 11 (5), Sept/Oct 1999, 779 – 797.

Junginger, S. (1998). Quantitative Bewertung von Geschäftsprozessmodellen: Eine Gegenüberstellung von rechnerischer Auswertung und Simulation. BPMS-Bericht, Vienna.

Perzel, K., Kane, D. (1999). Usability Patterns for Applications on the World Wide Web. *Proceedings of the PloP '99 Conference.*

Randa, P. (2001). Analysis Patterns in the Area of Administration. working document, University of Vienna, Dept. of Computer Science and Business Informatics.

Rossi, G., Schwabe, D., Lyardet, F. (1998). Patterns for Designing Navigable Information Spaces. *Proceedings of the PloP '98 Conference.*

Rumbaugh J., Jacobson I., Booch G. (1999). The Unified Modeling Language Reference Manual, Addison-Wesley.

Wohed P. (2000). Tool Support for Reuse of Analysis Patterns. Proceedings of the ER2000 Conference, Salt-Lake City, Oct. 2000, Springer-Verlag; Berlin-Heidelberg.