Complex Systems Informatics and Modeling Quarterly
CSIMQ, Issue 8, September/October 2016, Pages 49-59
Published online by RTU Press, https://csimq-journals.rtu.lv
https://doi.org/10.7250/csimq.2016-8.04
ISSN: 2255-9922 online

Managing Complexity in Activity Specifications by Separation of Concerns and Reusability

Peter Forbrig and Gregor Buchholz

University of Rostock, Albert-Einstein-Str. 22, 18051 Rostock, Germany

{peter.forbrig, gregor.buchholz}@uni-rostock.de

Abstract. The specification of activities of the different stakeholders is an important activity for software development. Currently, a lot of specification languages like task models, activity diagrams, state charts, and business specifications are used to document the results of the analysis of the domain in most projects. The paper discusses the aspect of reusability by considering generic submodels. This approach increases the quality of models. Additionally, the separation of concerns of cooperation and individual work by subject-oriented specifications is discussed. It will be demonstrated how task models can be used to support subject-oriented specification by so called team models and role models in a more precise way than S-BPM specifications. More precise restrictions on instances of roles can be specified.
Keywords: Activity specification, business process modeling, workflow specification, subject-oriented BPM, subject-oriented task models, cooperative task execution.

1 Introduction

Modeling seems to become more and more important for the implementation of interactive systems. Specification languages like UML or BPMN are very important for academia and industry. This is reflected by a lot of books and training courses. Model-based and model-driven software development methods have been established during the last decades.

From our point of view, modeling is also an attempt to reduce the complexity of software development. Based on this assumption we will discuss ideas of reducing the complexity of the development by organized reuse of already specified models.

Task models are discussed for this purpose. They have been proven to be useful as specification languages of functional requirements in Human Computer Interaction (HCI) (e.g., [1], [2], [3], and [4]). Additionally, they were used to support smart environments [5] and workflow management systems [6]. We will show that they also can be used to support the subject-oriented approach in business process modeling.

The subject-oriented approach can be characterized by the separation of models for different subjects and a joined communication model.

The paper is structured in the following way that after this short introduction reuse of models by generic submodels will be discussed. Afterwards, the advantages of the subject-oriented approach will be stated. We will describe and exemplify how our extension task models specify business processes more precisely than subject-oriented BPM (S-BPM) [7]. The paper will close with a summary and the idea of combining subject-oriented modeling with generic submodels. In this way separation of concerns can be combined with reuse.

2 Reuse of Models

Reuse of submodels is one established way of reducing the complexity of specifications. It can be considered as a special kind of separation of concerns. Certain aspects are specified in the submodel, thus, reducing the complexity of the model that reuses the submodel.

The following two paragraphs will recall some ideas that were presented in [8].

2.1 Reuse of Task Models

The term “task pattern” was discussed the first time by Breedvelt-Schouten et al. in [1]. The term “pattern” was used, but it referred only to a sub-tree. This kind of reuse is called submodel in the HAMSTERS environment ([2], [3]).

In [9] the usage of generic task patterns was suggested. A corresponding tool support was discussed in [10]. The adaptation of task patterns to a specific context of use and the insertion of the corresponding result into a large model was shown.

However, it was considered to perform parameter substitution during design time only. Indeed, to the best of our knowledge, in [8] the runtime substitution of parameter values was discussed the first time for such models. It was suggested to use the notation of HAMSTERS for such patterns. The notation allows specifying submodels, procedures, and conditional sub-trees.

An example that was given as introduction to BPMN modeling in [11] will be used to get an impression of this notation of HAMSTERS. The example specifies that somebody is recognized to be hungry. To prepare a meal groceries have to be acquired. After preparing the meal, it can be eaten and the hunger is satisfied.

In HAMSTERS pre- and post-conditions are not visually represented. Therefore, the task model looks like the graphics presented in Figure 1.

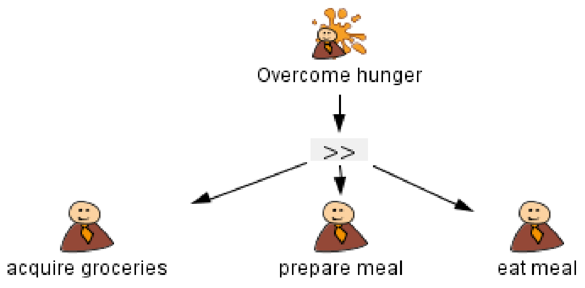


Figure 1. Task model for overcoming hunger

Most of the time, models can be generalized. The presented model could be generalized in such a way that activities of thirsty people are covered as well. The following generic task model is such a generalization (Figure 2). It fits for thirsty and hungry people. Additionally, different meals and drinks can be used in the specification. It can be instantiated to models for hungry and thirsty people.

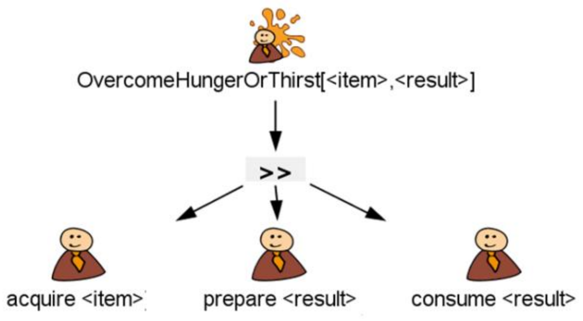


Figure 2. Generic component to overcome hunger or thirst

Two parameters are used. The first one is called “item” and can have the values “groceries” for meals, and “ingredients” for drinks. The second parameter “result” can be assigned to “meal”, “drink” or something else. Instantiating the task component with “groceries” and “meal” results nearly in the original model. The only difference is the task *eat meal* that is generalized to *consume meal* to provide an abstraction to *eat* and *drink*. A further parameter could have been introduced to deliver exactly the original model.

Additionally to simply propagating values, it is possible to make decisions based on these values during design time as well as during runtime. The latter is further distinguished into assignment at instantiation time and at decision/execution time. First experiments were performed with the design time parameter substitution [8]. In a case study existing task models describing tasks for a ground segment application were reengineered. The application is used to monitor and to control the Picard satellite that was launched in 2010 for solar observation. The original model consisted of 59 tasks. Restructuring the model reduced the number of specified tasks to 35. Readability was increased and even modeling mistakes were identified. The idea of generic submodels was later applied to workflow models.

2.2 Reuse of Workflow Models

20 workflow patterns were introduced by van der Aalst et al. in [12]. Most of these patterns describe knowledge on a low level of abstraction (see also [13], [14]). They describe solutions for *workflow patterns* on the level of language features like sequences, alternatives, procedure, metaphors, etc.

They are not on the level of describing something like buying a book in an e-shop or managing products in a store. In [15] we introduced the concept of generic components for BPMN specifications.

The hungry people example that was already used for task models [11] is specified in BPMN. Figure 3 provides this example that is abstracted in the specification of Figure 4.

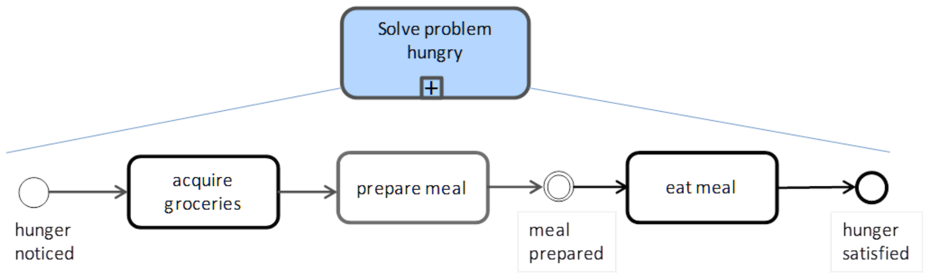


Figure 3. Workflow specification for hungry people

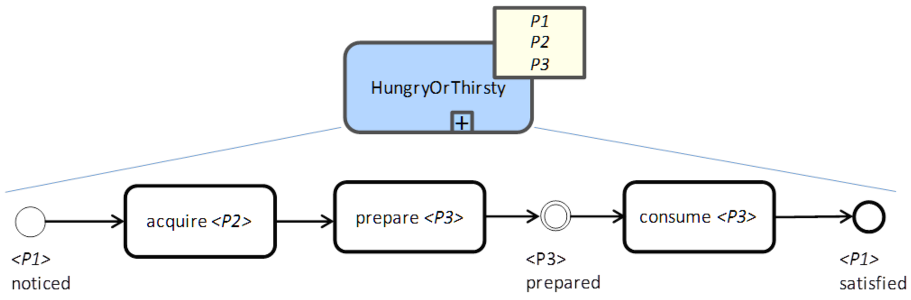


Figure 4. Generic workflow component for supporting hungry or thirsty people

Parameter P1 is expected to have the values “thirst” or “hunger”. The second parameter P2 specifies the required resources. It can have the values “groceries” or “ingredients”. The third parameter P3 specifies the wanted final result. The values could be “drink” or “meal”. However, they can also be more concrete like “pizza” or “Manhattan”.

To instantiate a generic component, values have to be assigned to the formal generic parameters. Such instances can be written like shown in Figure 5.

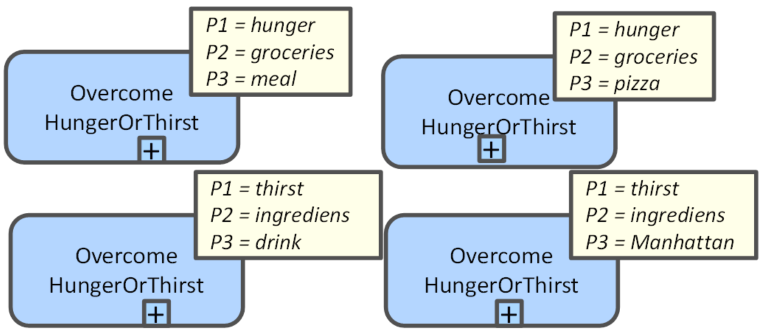


Figure 5. Instances of generic components

Figure 4 presents the component with parameter values “hunger”, “groceries”, and “meal” and the corresponding instance of the pattern with substituted parameters.

The concept was shown in [15] for more complex BPMN examples like delivering a pizza with different actors. It allows the reuse of specifications on different levels of abstractions and can help to reduce the complexity of models. The modeler can concentrate on the main aspects of the domain that have to be specified.

In [16] the idea of generic submodels for subject-oriented business process modeling in S-BPM is discussed as well. We will discuss the subject-oriented approach in the next paragraph because it has the advantage of separating the specification of large-scale communication and small-scale behavior specification. In other words, cooperation and individual work are specified separately.

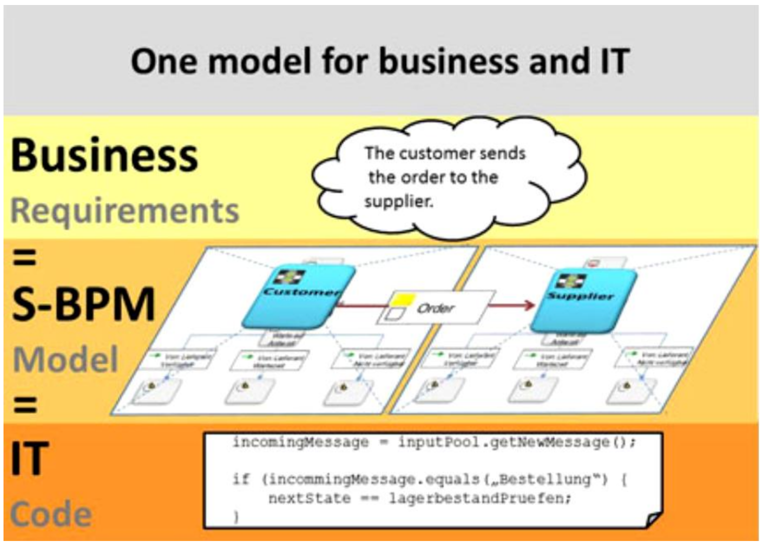
3 Subject-Oriented Business Process Modeling

Fleischmann et al. [17] provided an approach that allows the rapid execution of business process specification. It is called Subject-oriented Business Process Modeling and is supported by a language that is called S-BPM [7]. This approach perfectly fits to the idea of separation of concerns. There is no unified model of all activities but an overview by a communication model and separated process models for all subjects involved. These subjects are most of the time humans.

For S-BPM it is suggested that one starts with modeling the communication of subjects via messages first. In this way, the big picture is specified. Later, the dynamic behavior of each subject is specified by finite automata. In Figure 6 the approach is visualized. The available symbols of S-BPM are presented in Figure 7.

Requirements are the basis of a communication model that itself is the basis for the implementation, which is called IT Code in the Figure 6. There are only two subjects in Figure 6. They also exchange only one message. The subject *Customer* sends an *Order* to subject *Supplier*.

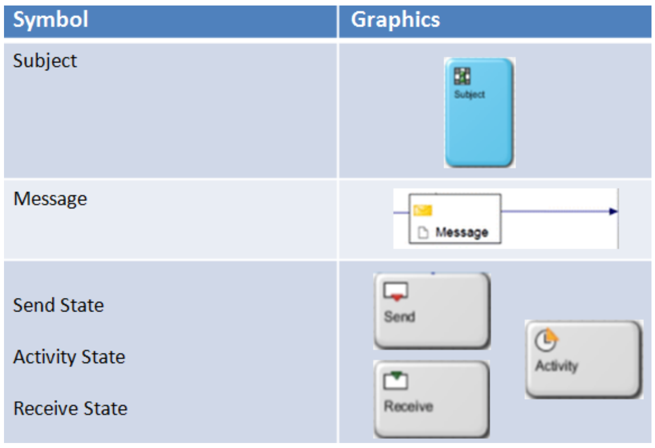
The details of the behavior of the subjects are defined by specific models. These models specify how they react on messages and under which condition subjects send messages.


One model for business and IT
Business
Requirements
=
S-BPM
Model
=
IT
Code
The customer sends the order to the supplier.
Customer
Order
Supplier

```
incomingMessage = inputPool.getNewMessage();

if (incommingMessage.equals(„Bestellung") {
    nextState == lagerbestandPruefen;
}
```

Figure 6. S-BPM approach (from https://www.metasonic.de/en/s-bpm)



Symbol	Graphics
Subject	Subject
Message	Message
Send State Activity State Receive State	Send Activity Receive

Figure 7. S-BPM notation (from [18])

We will recall an example from [18]. It is based on the following natural language description: *"An employee fills in a holiday application form. He/She puts in a start and end date of his/her vacation. The responsible manager checks the application and informs the employee about his/her decision; the holiday request might be rejected or get approved. In case of approval the holiday data are sent to the human resource department which updates the days-off in the holiday file."* ([18], p. 220). The message exchange between subjects is shown in Figure 8.

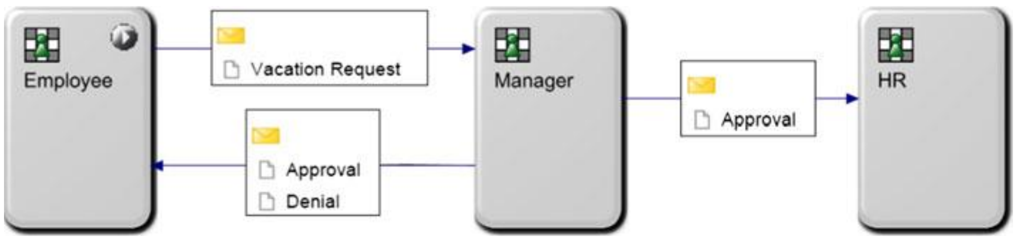


Figure 8. Example of message exchange between subjects (from [18])

The specification of the behavior of an employee is presented in Figure 9. It provides details of the business process from the perspective of the subject *Employee*. It specifies the behavior of an employee. This includes the communication with the subject *Manager*. It can be seen from this specification that a function in a function state (e.g., *Fill out Vacation request Form*) leads automatically to a message that informs about the ending of the performance of the function (e.g., “Fill out Vacation request form done”) and this leads to the action of “Sending it”. This action results into a message being sent to the manager.

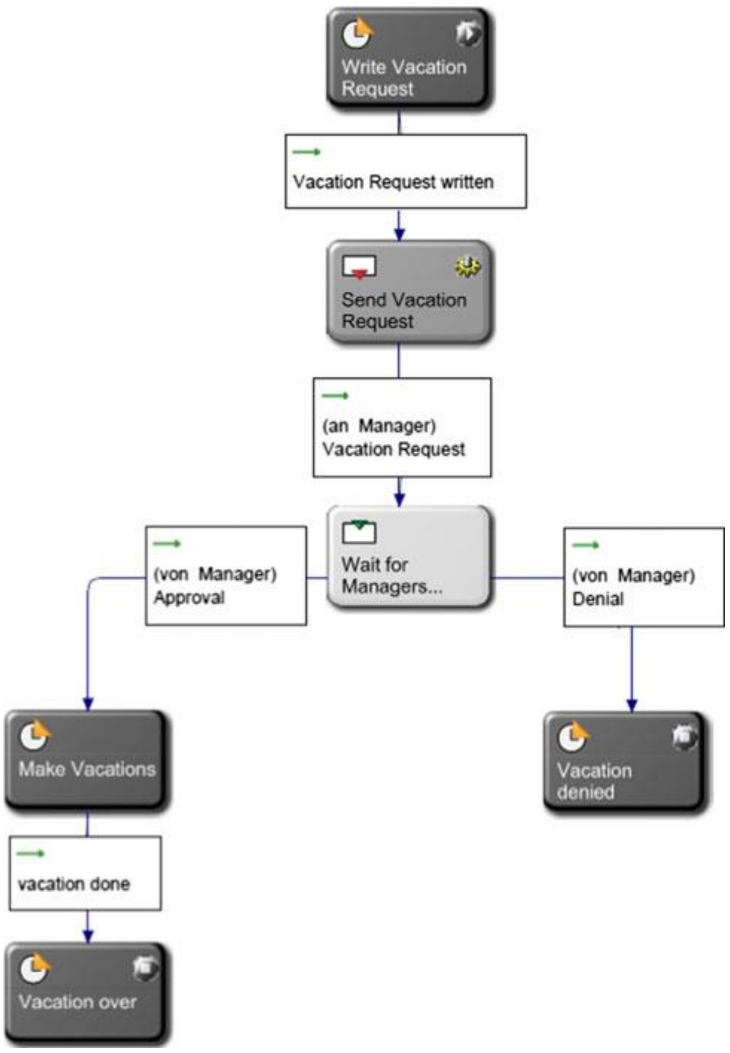
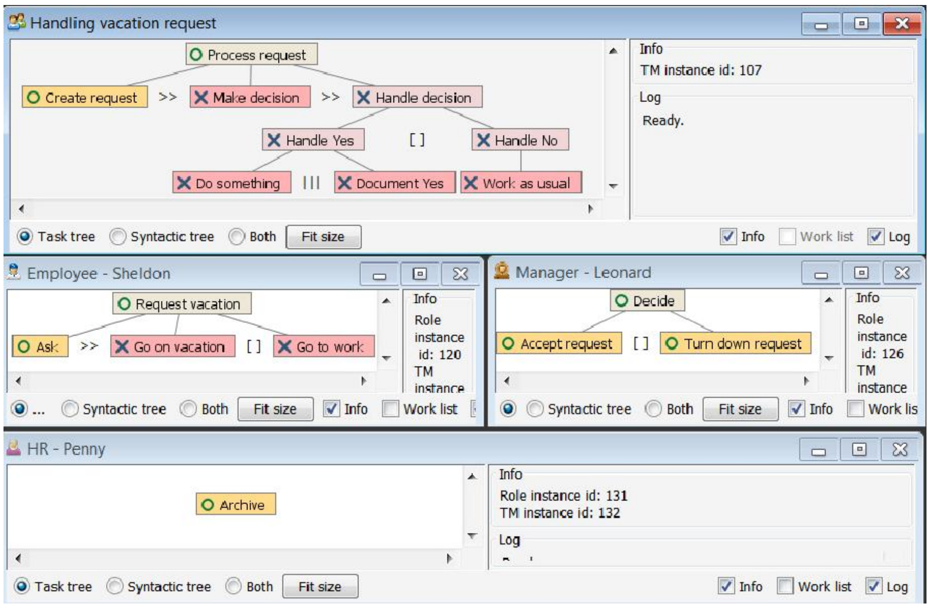


Figure 9. Example specification of the behavior of subject *Employee* (from [18])

A S-BPM specification like that in Figure 9 looks as a combination of the notation for EPCs [19] and for BPMN [11]. It drastically reduces the number of elements of BPMN and is easy to read. S-BPM specifications provide an interesting combination of overview diagrams and detailed diagrams. The detailed specifications are considerably reduced in their complexity in this way. However, there is no need to always use the S-BPM notation. Recently, Fichtenbauer et al. [20] reported about using BPMN for the subject-oriented approach. They restricted BPMN and made in this way the specifications ready for execution. This inspired us to use task models like discussed in Section 2 in the same way.

Task models have been used for several years to develop interactive systems [4]. We presented ideas to use task models for workflow systems [6] and we used task models for providing assistance in smart environments [21]. For this purpose, the specification language CTML was developed [5]. It consists of a team model, specifying the joined activities of a “team” in a smart environment, certain task models of stakeholders (described as roles), and state machines for the specification of the behavior of devices.

The team model was further refined to support the subject-oriented approach. The notation is called CoTa (Cooperative Tasks). In Figures 10–13 we will show how the team model can play the role of the communication model and how task models describe the behavior of subjects.



Info
TM instance id: 107
Log
Ready.

Info
Role instance id: 120
TM instance

Info
Role instance id: 126
TM instance

Info
Role instance id: 131
TM instance id: 132
Log

Figure 10. Team model instance and role model instances

The team model is the upper part of Figure 10. It specifies that for processing a vacation process such a request has to be created first. Afterwards, a decision has to be made and, finally, the decision has to be handled. This can be done in two ways. Either a positive or a negative decision has to be handled. In the positive case, the employee can have holidays and relax. In parallel, the vacation has to be documented. In the negative case, the employee has to continue her/his work.

The middle part of Figure 10 presents the task model for the role Employee. There can be several instances of such a model. Currently, the instance for Sheldon is shown. Leonard performs the role Manager. Sheldon has first to ask, afterwards she can go on vacation or go to work. Leonard can accept or turn down the request.

The lower part is the simple model for the role HR. Penny has to archive the vacation request only. These models are already presented in an animated mode. The green cycle signals that the corresponding task can be started. Leafs of animated instances of role models can be executed only. One cannot interact with the team model. Its state changes are based on the other models. They provide the necessary events and conditions. However, the team model can restrict other models. A certain task may not be able to be executed while the team model is in a certain state.

Task trees can be visualized as task trees or syntactic trees like task models were introduced with the HAMSTERS tool. It is also possible to present both versions together. Figure 11 presents the team model as a syntactic tree. It has the advantage that there is no need to know the priority of temporal relations. However, there is the disadvantage of having tasks on different hierarchy levels that are otherwise on the same level.

The start trigger of the team task Create request is Employee.oneInstance.Ask.start, which means that it is started when any instance of Employee started Ask. The end trigger might be Employee.oneInstance.Ask.end. The language for the conditions is a kind of simplified predicate logic in the notation of Object Constraint Language (OCL). Currently, one can specify oneInstance or allInstances in the conditions. According to our experience, there is no urgent need for more expressive expressions.

The specified conditions for the team model fit to our mental model if only one instance of the role Employee exists. Otherwise, start and end could be executed by different instances. This might not be the intention of the modeler. Therefore, the idea of binding context and references to role instances was introduced. The first time an instance provides a trigger, its identification can be stored in a variable.

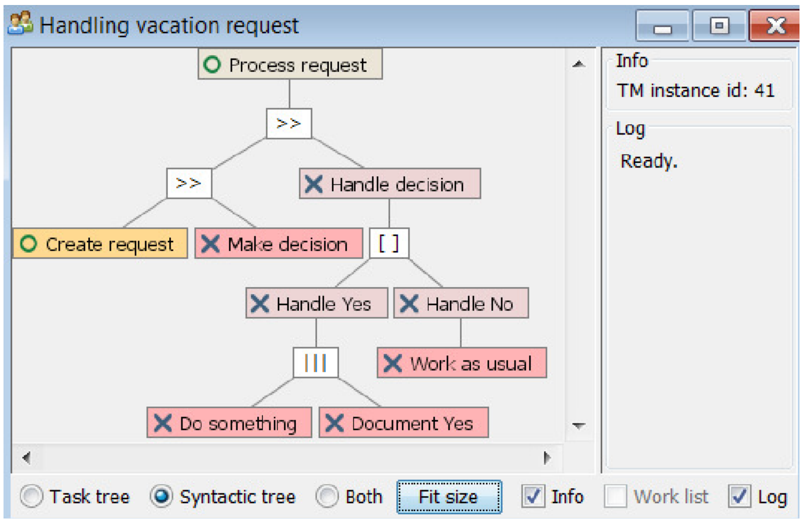


Figure 11. Visualization of the team model as a syntactic tree

Assuming that for role Employee the variable E1 is defined, the conditions can change to E1.Ask.start and E1.Ask.end. In this case, the same instance is referred to. The same can be specified for the role Manager. The same manager that starts the decision has to end it. However, this might not be necessary. Any other manager might have the right to end the decision. For an employee it might be fine if any manager gives the okay.

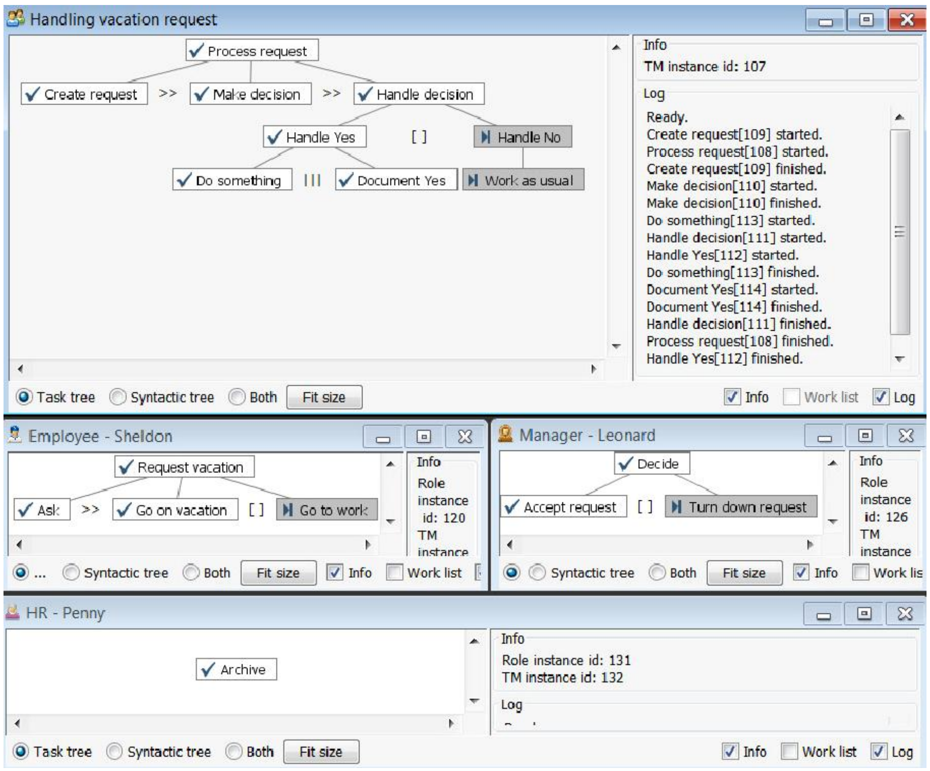
Nevertheless, the employee that is allowed to go on vacation has to be the same as that who asked for vacation. These details can be seen in the specification below. This is a more precise specification than the S-BPM model presented above. S-BPM does not allow specifying such details.

It is simply assumed that always the same instances are involved. However, this might not always be the case.

The main part of the team model looks like this:

```
<task name="Process request" bindingContext="C1">
  <roleVar role="Employee" var="E1"/>
  <roleVar role="Manager" var="M1"/>
  <roleVar role="HR" var="H1"/>
  <task name="Create request" operator="enabling"
    startTrigger="E1.Ask.start" endTrigger="E1.Ask.end" />
  <task name="Make decision" operator="enabling"
    startTrigger="M1.Decide.start" endTrigger="M1.Decide.end" />
  <task name="Handle decision">
    <task name="Handle Yes" operator="choice">
      <task name="Do something" operator="interleaving"
        startTrigger="E1.GoOnVacation.start"
        endTrigger="E1.GoOnVacation.end">
      <bindPrec source="M1.AcceptRequest" target="E1.GoOnVacation" />
      </task>
      <task name="Document Yes" startTrigger="H1.Archive.start"
        endTrigger="H1.Archive.end" />
    </task>
    <task name="Handle No">
      <task name="Work as usual" startTrigger="E1.GoToWork.start"
        endTrigger="E1.GoToWork.end" />
    </task>
  </task>
</task>
```

The state of the model instances after a positive decision of the manager can be seen in Figure 12. The performed scenario is logged at the right side of the team model. It is also visualized which tasks were executed and which were skipped.



```
Info
TM instance id: 107
Log
Ready.
Create request[109] started.
Process request[108] started.
Create request[109] finished.
Make decision[110] started.
Make decision[110] finished.
Do something[113] started.
Handle decision[111] started.
Handle Yes[112] started.
Do something[113] finished.
Document Yes[114] started.
Document Yes[114] finished.
Handle decision[111] finished.
Process request[108] finished.
Handle Yes[112] finished.
```

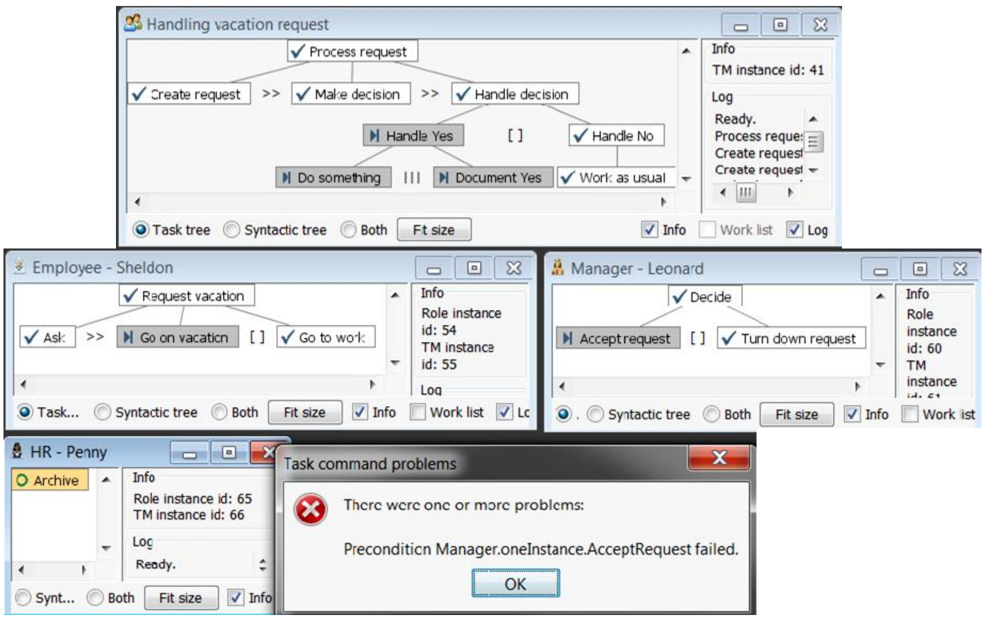
```
Info
Role instance id: 120
TM instance
```

```
Info
Role instance id: 126
TM instance
```

```
Info
Role instance id: 131
TM instance id: 132
Log
```

Figure 12. State of the task model after a positive manager decision

The models in Figure 12 have reached their final states. There is no further task that can be executed. The corresponding business process has ended. In case Process request is specified as instance iteration, a new instance can be created. This can even be done at any time.



```
Info
TM instance id: 41
Log
Ready.
Process reque
Create request
Create request
```

```
Info
Role instance id: 54
TM instance id: 55
Log
```

```
Info
Role instance id: 60
TM instance
```

```
Info
Role instance id: 65
TM instance id: 66
Log
Ready.
```

```
There were one or more problems:
Precondition Manager.oneInstance.AcceptRequest failed.
```

Figure 13. Final model instance states after a rejected request

Several instances can run in parallel. In this case, the binding of communication partners becomes important.

There is, of course, an alternative path possible. The manager can turn down the request. In this case, the result of the business process will be that of Figure 13. The role HR was not involved in the execution of the tasks. In case one wants to execute the corresponding task, an error window will pop up and will show the condition that is not fulfilled. No manager exists that accepted a request. This condition is currently specified in the HR model. This could be done more precisely in the team model as well.

Recently, the interpreter for the task models was implemented in a cloud. In this way task models can be executed via the Internet in an cooperative way. This allows a subject-oriented development of interactive systems for different platforms.

4 Conclusion

Two different aspects of coping with complexity were discussed. These aspects are reuse of generic submodels of activities and subject-oriented modeling of business processes. Both are related to the idea of separation of concerns. Separate models help to focus on specific details without the need of looking at the global relations all the time. Generic submodels, sometimes even called patterns, and role models together with a team model specifying the communication between the different models might be helpful.

Subject-oriented specification was the second aspect. It allows the separation of global message exchange from local behavior of subjects as the result in reaction of received messages. Specific task models supporting the subject-oriented specification of business processes were introduced. The expressiveness of this approach was shown by a simple example from the subject-oriented community research.

We were able to create a specific task-model notation CoTa (Cooperative Tasks) in XML that allows more detailed relations between different model instances. This kind of specifications is more precise than the general business process specifications like, e.g., in S-BPM.

The corresponding tool support for an interpreter CoTaSE (Cooperative Task Simulation Environment) was implemented in Java. Recently the implementation of the task model interpreter was adapted for a cloud. This allows the creation of collaborative applications like workflow systems.

The integration of generic submodels into the tool support is one challenge for the future. It should be combined with ideas of reuse for S-BPM like presented in [20]. A further cross-pollination of ideas from S-BPM and CoTa seems to be possible and useful. This should be combined with further case studies.

Currently, CoTaSE is integrated into a smart meeting room at one of the departments at the University of Rostock. It is also part of a specification tool running on an interactive table to specify test scenarios for the smart meeting room. It allows tangible interaction and an evaluation of the specified model in a virtual environment.

The extended abstract of this paper is available in [22].

References

[1] M. Breedvelt-Schouten, F. Paternò and C. Severijns, “Reusable Structures in Task Models,” in Proc. the DSV-IS 1997, pp. 225–239, 1997. Available: http://dx.doi.org/10.1007/978-3-7091-6878-3_15

[2] C. Martinie, P.A. Palanque and M. Winckler, “Structuring and Composition Mechanisms to Address Scalability Issues in Task Models,” in Proc. the INTERACT 2011, vol. 3, pp. 589–609. Available: http://dx.doi.org/10.1007/978-3-642-23765-2_40

[3] C. Martinie, P. Palanque, M. Ragosta and R. Fahssi, “Extending Procedural Task Models by Explicit and Systematic Integration of Objects, Knowledge and Information,” in Proc. the ECCE 2013, no. 23, pp. 1–10, 2013. Available: http://dx.doi.org/10.1145/2501907.2501954

[4] A.R. Puerta, “A Model-Based Interface Development Environment,” IEEE Software, vol. 14, no. 4, pp. 40–47, July/Aug. 1997. Available: http://dx.doi.org/10.1109/52.595902

[5] M. Wurdel, D. Sinnig and P. Forbrig, “CTML: Domain and Task Modeling for Collaborative Environments,” in Journal of Universal Computer Science, vol. 14, no. 19, pp. 3188–3201, 2009.

[6] J. Brüning and P. Forbrig, “TTMS: A Task Tree Based Workflow Management System,” in T. Halpin, S. Nurcan, J. Krogstie, P. Soffer, E. Proper, R. Schmidt, I. Bider, Eds., BPMDS 2011 and EMMSAD 2011, LNBIP, vol. 81, Springer, Heidelberg, pp. 186–200, 2011. Available: http://dx.doi.org/10.1007/978-3-642-21759-3_14

[7] A. Fleischmann, W. Schmidt and C. Stary, “Open S-BPM= Open Innovation,” in S-BPM ONE-Running Processes, Springer Berlin Heidelberg, pp. 295–320, 2013. Available: http://dx.doi.org/10.1007/978-3-642-36754-0_19

[8] F. Forbrig, C. Martinie, P. Palanque and M. Winckler, “Rapid Task-Models Development Using Submodels, Sub-routines and Generic Components,” in Proc. the HCSE 2014, LNCS, vol. 8742, pp. 144–163, 2014. Available: http://dx.doi.org/10.1007/978-3-662-44811-3_9

[9] A. Gaffar, D. Sinnig, A. Seffah and P. Forbrig, “Modeling Patterns for Task Models,” in Proc. the 3rd annual conference on Task models and diagrams (TAMODIA '04), ACM, New York, NY, USA, pp. 99–104, 2004. Available: http://dx.doi.org/10.1145/1045446.1045465

[10] F. Radecke and P. Forbrig, “Patterns in Task-Based Modeling of User Interfaces,” in Lecture Notes in Computer Science, vol. 4849, pp. 184–197, 2007. Available: http://dx.doi.org/10.1007/978-3-540-77222-4_15

[11] BPMN. [Online]. Available: http:/www.bpmn.org: (accessed 03.02.2016)

[12] W.M.P. Van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski and A.P. Barros, “Workflow Patterns,” in Distributed and Parallel Databases, vol. 14, no. 3, pp. 5–51, July 2003. Available: http://dx.doi.org/10.1023/A:1022883727209

[13] S.A. White, “Process Modeling Notations and Workflow Patterns,” BP Trends, IBM Corporation, pp. 1-24, March 2004. [Online]. Available: http://www.bptrends.com/publicationfiles/03-04%20WP%20Notations%20and%20Workflow%20Patterns%20-%20White.pdf (accessed 22.11.2015)

[14] P. Wohed, W.M.P. van der Aalst, A.H.M. ter Hofstede and N. Russel, “On the Suitability of BPMN for Business Process Modelling,” in Lecture Notes in Computer Science, vol. 4102, pp. 161–176, 2006. Available: http://dx.doi.org/10.1007/11841760_12

[15] P. Forbrig, “Generic Components for BPMN Specifications Perspectives,” in Proc. the Business Informatics Research – 13th International Conference, BIR 2014, Lund, Sweden, Sept. 22–24, 2014, pp. 202–216, 2014. Available: http://dx.doi.org/10.1007/978-3-319-11370-8_15

[16] P. Forbrig, “Reuse of Models in S-BPM Process Specifications,” in Proc. the 7th International Conference on Subject-Oriented Business Process Management, S-BPM ONE 2015, Kiel, Germany, Apr. 23–24, 2015, pp. 6–16, 2015. Available: http://dx.doi.org/10.1145/2723839.2723846

[17] A. Fleischmann, W. Schmidt and C. Stary, “Requirements Specification as Executable Software Design – A Behavior Perspective,” in R. Matulevičius et al., Eds., REFSQ Workshop proceedings, pp. 9–18, 2015. [Online]. Available: http://ceur-ws.org/Vol-1342/01-CRE.pdf

[18] A. Fleischmann and C. Stary, “Whom to Talk to? A Stakeholder Perspective on Business Process Development,” in Universal Access in the Information Society, vol. 11, no. 2, pp. 125–150, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10209-011-0236-x

[19] Event-Driven Process Chain. [Online]. Available: http://en.wikipedia.org/wiki/Event-driven_process_chain (accessed 03.05.2016)

[20] Ch. Fichtenbauer and A. Fleischmann, “Three Dimensions of Process Models Regarding Their Execution,” in Proc. the 8th International Conference on Subject-Oriented Business Process Management (S-BPM '16). ACM, New York, NY, USA, no. 7, p. 8. Available: http://dx.doi.org/10.1145/2882879.2882892

[21] M. Zaki and P. Forbrig, “Making Task Models and Dialog Graphs Suitable for Generating Assistive and Adaptable User Interfaces for Smart Environments,” in Proc. the PECCS 2013, Barcelona, Spain, pp. 66–75, 2013. Available: http://dx.doi.org/10.5220/0004315100660075

[22] P.Forbrig and G.Buchholz, “Managing Complexity in Activity Specifications by Separation of Concerns (extended abstract),” B. Johansson, F. Vencovský, Eds., Joint Proceedings of the BIR 2016 Workshops and Doctoral Consortium co-located with 15th International Conference on Perspectives in Business Informatics Research (BIR 2016), Prague, Czech Republic, September 14–16, 2016. [Online]. Available: http://ceur-ws.org/Vol-1684/paper17.pdf