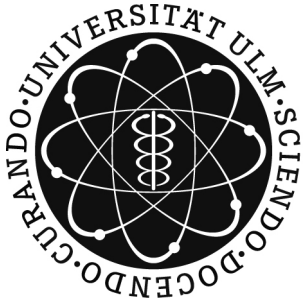
Specification and Management of Complex Resources in a Process Environment

Masterarbeit an der Universität Ulm
Fakultät für Informatik



vorgelegt von
Holger Bähren

Gutachter:
Prof. Dr. Manfred Reichert, Universität Ulm
Prof. Dr. Peter Dadam, Universiät Ulm

Betreuer:
Andreas Lanz

Oktober 2012

Table of Contents

Abstract

Business process management system have become very important in the last years. They have changed many different domains. One area where so far very little interaction with processes has taken place is in the specification and management of resources. The high importance that resources have in organizations and their direct influence on the execution of processes, show, however, that this is an ideal subject to bring into the business process world.

This work shows how a model for the specification and management of complex resources in a process environment can be developed. It demonstrates what kind of requirement have to be met, both for the sake of correct representation of the resources themselves and for the sake of a functioning integration with the business processes. It shows how all aspects of the business processes have to be taken into account and what concepts are important in the different phases of the business process lifecycle. The impact that the integration of resources has on adaptive process management is presented. Similarly it is shown what business intelligence can achieve for resources. Finally, a working prototype for the integration of resources into the business process management system AristaFlow is shown.

1 Introduction

The use of software to support the organization and day-to-day work of businesses and other organizations has been one of the greatest disruptions to the commercial world in the last decades. From multinational corporations to small family businesses there has been a fundamental change in the way products and services are produced, sold and distributed. Software support has made possible many, completely new, products and has given companies the opportunity to produce cheaper, faster, and more efficiently.

1.1 Motivation

Since the industrial revolution the focus of work has shifted from the individual skill of the craftsman to finding a way, so that all the different parts of a business, the employees, machinery, raw materials, and manufacturing sites all work together to create the final product.

The efficient allocation of resources is therefore one of the cornerstones of a successful organization. Idling resources, resource conflicts and misapplied resources weaken a business and can lead to dissatisfaction in the work force as well as a substandard product. Consequently, the knowledge where a resource is needed most, what resources work together best and how shortages can be prevented, is some of the most important information that exists within an organization.

Unfortunately, most of this information is currently locked away in the heads of the people that work with these resources on a daily basis. Coordination of resources often does not happen on the basis of structured and understandable rules, but by ad-hoc decisions of many isolated decision makers.

For a long time, a similar problem has existed with the different work flows in organizations. The information about which steps are taken in sequence and which individual operation should be performed next existed only in the heads of those people directly involved. Business process management has succeeded in bringing the implicit workflows to light and allowing them to be analyzed and improved on. Especially with the advent of business process management systems, it has become evident how the efficiency of organizations can be improved by a process-centric approach.

It is therefore only logical to ask, if the efficiency of resource management can be enhanced in the same way by business process management. If business process management can help give the same structure to the way resources are handled inside organizations, then it

will be possible to eliminate a great amount of waste and disruption from the work process in a great number of different domains.

1.2 Aim and Thesis Requirements

The aim of this thesis is to develop a model by which the specification and management of complex resources can be achieved in a process environment. First it is necessary to work out and understand all the requirements that such a model has to fulfill. Both the field of resource management and the domain of business process management have to be analyzed to recognize the connection points between the two. For this purpose, the many different ways in which resources are used have to be taken into account. In the same way the thesis has to consider the whole complexity of business process management systems.

One of the integral parts of resource management is the underlying meta model that characterizes the resources. Here it is important to strike a balance between two different requirements. On the one hand, the meta model has to be flexible enough to be applicable in a wide range of different domains. On the other hand, the meta model has to be easy to use for designers, users and administrators that work with the software. When designing this model, it is very important to keep in mind the characteristics of a process environment.

Very important for the model is also the system used to specify which resources or resource combinations are required for the execution of every process step. This specification should be both expressive and easy to use. It has to be able to work with the great number of different resources that exist in any large organization.

The model should be independent from any specific business process management system. It should be possible to implement the model for any mainstream business process management system. The model has to take into account the real problems that occur when working with resources in a process environment. The role of resources in all different parts of the business process lifecycle [Wesk07] have to be analyzed.

Finally a prototype of the developed model should be implemented that extends an existing business process management system with a resource management perspective.

1.3 Thesis Structure

This thesis consists of the following essential parts. In chapter 2 the general requirements for a system that integrates resource and process management are described. For this purpose first an overview of business process and resource management is given. Then the individual requirements are presented in detail. The following chapter 3 presents related works and technologies. This includes different resource data models and query languages like ClassAd, Redline, and RDF/SPARQL. Furthermore a number of existing solutions for the integration of resource and process management are analyzed. Chapter 4 describes how resources have to be specified and classified to work together with a process management system and what further steps have to be taken in the process modeling phase. Chapter 5

shows how resources have to be integrated during process execution. Chapter 6 takes a closer look at the interplay of resources and processes in regard to process monitoring, logging and business intelligence. Chapter 7 presents a prototypical implementation of a system that integrates resource and process management on the basis of ADEPT2/AristaFlow. Chapter 8 finally, offers a summery of the results.

2 General Requirements for Managing Complex Resources in a Process Environment

Managing resources has always been an important part in every enterprise, but for a long time this has been given relatively little attention in the context of business process management. While business process management has gained importance and found its way into many different fields of application, resource management has lagged behind. Although some business process management systems support a rudimentary resource model many of the complexities that occur in the interplay between resources and processes in a real organization are omitted.

In the same way, modeling languages don't take into account the real problems of interacting with resources. "Even in widely-used process modeling languages such as BPMN and EPC, complex resource interactions between activities and resources are ignored. For instance, BPMN models only allow for modelling pools and lanes to define the roles of the resources that may carry out an activity (...) As a result, many Business Process Management Systems only cater for simple one-to-one resource allocation strategies" [OWF+10].

This chapter gives a short overview of business process management in chapter 2.1 and of resource management in chapter 2.2. In chapter 2.3 the requirements for an integration of business process management and resource management are developed in detail.

2.1 Overview Business Process Management

The foundation of *business process management* is the fact that every product and service a company produces can be traced back to a number of activities that are performed in the company [Wesk07]. If the people and other resources included in these activities work together well the business goals of the company can be reached. If not, disorganization, duplicate work processes and dissatisfied employees are the result and the organization suffers. Information technology can be a tremendous help in the organization, execution and monitoring of these activities.

Since much of this thesis uses business process terminology, it is helpful to define a few key concepts. Weske in [Wesk07] defines a *business process* as "a set of activities that are performed in coordination in an organizational and technical environment. These activities jointly realize a business goal. Each business process is enacted by a single organization, but it may interact with business processes performed by other organizations."

2 General Requirements for Managing Complex Resources in a Process Environment

A *business process management system* in turn is a "generic software system that is driven by explicit process representations to coordinate the enactment of business processes." [Wesk07].

A *business process model* "consists of a set of activity models and execution constraints between them" [Wesk07] while a *business process instance* "represents a concrete case in the operational business of a company, consisting of activity instances" [Wesk07].

At the core of business process management rests the *business process lifecycle*. As seen in Figure 1, the lifecycle of processes consists of four different phases, design and analysis, configuration, enactment and evaluation [Wesk07].

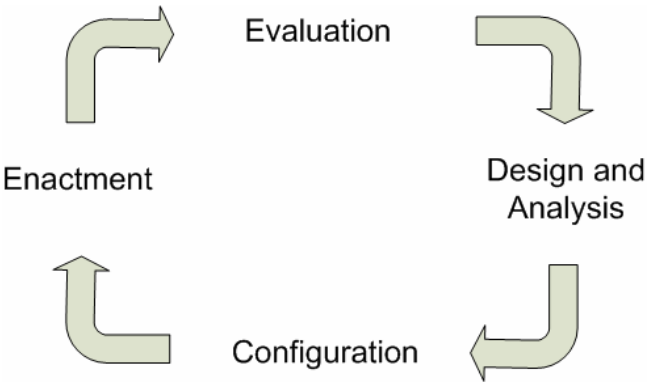


Figure 1: Business Process Lifecycle, based on [Wesk07]

In the *design and analysis phase* surveys are conducted to identify the actual business processes. The discovered processes are then modeled using a specialized business process modeling language. The modeled processes then need to be validated by using workshops and simulation techniques. Verification of the processes makes sure that the resulting process is correct and doesn't contain unwanted properties like deadlocks.

After the design and analysis phase the business process is implemented in the *configuration phase*. This does not necessarily need to take place with the help of a business process management system but can instead be done by a number of policies and routines. If a software system is used it needs to be configured so the organizational structure of the organization and the business process are represented correctly. The implemented program then has to be tested to make sure the process shows the expected behavior.

Next the instances of the business process can be *enacted*. This represents the actual runtime in which the process is executed. The execution of the process is controlled by the business process management system. Usually a monitoring component visualizes and

provides information about the current state of the running process instances. After the process instance has terminated execution data is used in the *evaluation phase* to analyze and improve the quality of the business process model.

Business process management can be used in a wide variety of contexts. Its flexibility allows the application in every field where the work is done in a structured and organized way. Contrary to its name it is not restricted to businesses. Public institutions and NGOs can profit from business process management in the same way.

2.2 Overview Resource Management

Resource management is not a separate field unto itself but rather is an important factor in a number of different domains. It is especially critical in the areas of project management, grid computing and the semantic web. Since all of these fields see resources in a different way it helps to examine resource management from these different viewpoints.

In *project management* resources are seen in two different contexts. *Resource planning* determines what kind of resources (people, equipment, materials) are needed for a specific project, in what quantity they should be used and when they should be present to perform certain tasks [PMBK08].To answer these questions it is necessary to know how the project is broken into smaller subprojects, what kind of resources were needed on similar projects in the past and what kind of resources are potentially available both inside and outside the organization. Organizational policies regarding the deployment of resources have to be taken into account as well [PMBK08].

Project human resource management on the other hand deals only with human resources, but defines this broadly as all people that are involved with the project. It includes organizational planning, staff acquisition and team development. "Organizational planning involves identifying, documenting, and assigning project roles, responsibilities, and reporting relationships" [PMBK08]. This is mostly done at the beginning of the project, since much of the later work and interplay between participants is dependent on project roles and responsibilities.

Staff acquisition describes how the human resources that are needed for the project are acquired and put together into teams. The concrete circumstances of this differ greatly depending on the project and organizational structure, but usually this includes negotiations with the "owners" of the needed resource (e.g. department heads) and a balancing which capabilities and prior work experience are needed the most. *Team development* tries to take the existing human resources and make sure that they work together toward the project goal. Thereby it tries to enhance the productivity both of the individual resources and of the team itself. This is achieved by the general management skills of the team leader, a reward system as well as training and team-building activities [PMBK08].

In contrast to this in *grid computing* a resource is seen as "a generic term to indicate a physical device, service, data item, or other entity for which discovery and selection procedures are required" [LiFo03]. The main focus is the efficient assignment of these

resources to the consumers. To achieve this, it is necessary to have a model of the existing resources which makes it possible for the agent requiring resources to have exact information about the abilities and availability of the resources [RLS98]. The efficiency of the assignment is measured by a predefined performance metric. Additionally the different resources can have different and complex usage policies that only allow certain agents access to the resources or limit the use of the resource at certain times.

In the context of the *semantic web* resources are anything that can be described by a *uniform resource identifier*. This can be web resources as well as persons and any other types of directly identifiable objects. The main objective is to organize and connect information about resources and make them readable by machines. This semantic structuring allows the automatization of complex tasks that have - up to now - been the domain of humans. Certain techniques and technologies like the Resource Description Framework (RDF) [RDF04] that have been designed for the semantic web can be used in a more general way for the direct managing and matching of resources and resource demands.

As seen in this chapter there are significant differences in the way resources are interpreted. Some of these are more suited to the integration with business process management than others. It is necessary to keep in mind the large spectrum represented by resources and include different parts of different resource definitions when designing the requirements for the integration of business process management and resource management.

2.3 Requirements

For an integration of resource management into business process management to work, there are a number of different requirements that have to be fulfilled. This chapter deals with the demands that are placed on the integrated system while the resulting requirements that concern the separate components of the system are discussed in their respective chapters.

1. Expressive resource definition and structure: Defining the way resources work and are handled is one of the key points of the integration. The resource structure has to include a number of different features to make sure that all situations that arise in a process environment are covered. In the realm of resource classification this means that all types of resources have to be covered.

   The system has to work with *human* and *non-human resource*. Since not every specific resource type can be hard-coded into the system there has to be a way for modelers to design the types of resources that will be used later. Therefore modelers have to be able to create both custom attributes for the resources and relations between different types of resources.

   These user-defined resource types must also be able to inherit attributes and

relationships from another class. This requires a structure that correctly describes the inheritance states of all resources.

2. Support for multiple resources: While some business process management systems have a rudimentary support for resource management, most of them only have the ability to link a single human resource with a process step [OWF+10]. For a integrated system that aims to map the complexities of real world business processes this falls significantly short.

   The system therefore has to support multiple resources for every process step. Additionally some resources can work on several process steps at the same time, so multitasking for resources has to be possible. Since resources are often used in combination only, the system must allow for groups of resources. These groups should be structured depending on their make up and should be usable in the same way that singular resources are.

3. Independence of the resource model from the process model: Since the design of the resources is not directly related to the process view, the structure of the resource types and their relations in an organization should not be dependent on the process model.

   An independent resource design allows the designer to concentrate on the resources themselves instead of having to keep the process model in mind for every step. This also facilitates easier integration into and switch between different business process management systems.

4. Flexible resource allocation: The correct allocation of resources to their respective process steps is subject to a lot of different influences and decisions. The system has to be able to map these influences correctly and make sure that the allocation takes place according to predefined rules and preferences.

5. Ease of use: Every step of the resource and process management should be as easy to use as possible. As many steps of the system as possible should be accomplished by graphical user interfaces. Especially for the basic functions there should be no requirement to learn new query or scripting languages.

   Modeling the resource structure and types should be possible without deep knowledge of the process model and the assignment of resources for a process step shouldn't require complete knowledge about the workings of the resource management.

6. Comprehensible monitoring and logging: All changes that are made to the resource allocation in the course of the process execution should be logged to allow for post-execution analysis and build a basis for process mining. Especially the allocation of rare resources to certain process steps is very suitable for a thorough analysis to find the most efficient and fastest way of execution. Monitoring and logging is also helpful to find flaws in the design of the resources and identify over- and undercapacities of certain resources.

2 General Requirements for Managing Complex Resources in a Process Environment

These requirements are the first part needed to develop and design an integrated system for resource and process management. Another part are the tools and technologies that are discussed in the following chapter.

3 Related Works and Technologies

To lay the groundwork for a working model for the specification and management of complex resources, it is necessary to first examine what kind of tools and technologies are available to identify and solve the occurring problems. As a first step it is important to find a data model that is suited to the specification of resources. Additionally a query language needs to be found that allows the interrogation and interaction with of the resource.

Chapter 3.1 illustrates *resource data models* and *query languages*. At first a general overview is given in Part 3.1.1. Chapter 3.1.2 introduces ClassAd, while Redline, the Resource Description Framework (RDF), and SPARQL are dealt with in Chapter 3.1.3, 3.1.4 and 3.1.5 respectively. Finally, in Chapter 3.2 a number of existing solutions for the integration of resource management and process management are analyzed.

3.1 Resource Data Models and Query Languages

Since resources are used in many different applications with different context it is no surprise that there exist a lot of different data models and query language for these resources. No domain has exactly the same requirements for resources and often there are complex interactions between the resources and other parts of the software that necessitate features for the resource data model that don't always make sense in a different context. Still, it is possible to give an overview of the most important and corresponding features that these data models and their languages have in common.

3.1.1 Overview

The *resource data models* are the basis of resource management. The structure of the data model decides what kind of resources and resource interactions may be specified and on which feature of the resources the focus lies. The structure of the model is very much based on the area in which it is supposed to be used. A data model for grid computing necessarily looks different to a data model for the semantic web.

The composition of the queries or the query language is also dependent on the data model. Query languages can only find resources based on to the features that they possess according to the data model. In some cases like ClassAd [RLS98] and Redline [LiFo03] the structure of the query is defined by the data model itself, so there does not exist any real distinction between the model and the query. In other cases like RDF [RDF04] the model

only defines the resources themselves and leaves the way in which they are matched up to the query language.

Additionally, not all query languages have the same set of features. While all are able to match a single resource to a request, there are certain requests that only some query languages can solve. Set-Matching, for example allows the user to request a set of resources that in combination fulfill a certain requirement, e.g., a set of computers that have an aggregate RAM of 64 GB. Gang-matching in turn allows a user to request several resources at once, but only on the basis of their individual attributes.

3.1.2 ClassAd

ClassAd is a "language for expressing and evaluating attributes" [RLS98] that is used extensively in the high-throughput computing framework Condor [ELD+96] developed by the University of Wisconsin-Madison. The resources in this system are relatively flexible and can be defined by the user. The system uses a "semi-structured data model - the classified advertisements data model - to represent the principals of the system and folds the query language into the data model, allowing entities to publish queries (i.e. requirements) as attributes" [RLS98].

The ClassAd framework consists of 5 basic components [RLS98]. The first component is the ClassAd specification. It "defines a language for expressing characteristics and constraints, and a semantics of evaluating these attributes" [RLS98]. The advertising protocol governs what has to be part of the ClassAd, to be recognized by the matchmaker and included in the matchmaking process. The matchmaking algorithm itself specifies "how the contents of ads and the state of the system relate to the outcome of the matchmaking process" [RLS98]. The fourth component is the matchmaking protocol. It defines what happens in the event of a match, it handles the notification of the matched entities and governs what information they receive. Finally the claiming protocol states what matched entities have to do to activate the service.

The ClassAd data model is an example of a semi-structured data model. Two different ClassAds of the same type can have different sets of properties, depending on the person modeling the ad. Only the attributes for which a match is supposed to be achieved have to be the same, differences in other parts of the ClassAd do not represent a problem.

Unfortunately ClassAd is very restrictive and has difficulties describing resources that don't consist of computing hardware. ClassAd can only perform exact matches using a predefined level of complexity and cannot cope with a wide array of differently phrased descriptions. Additionally both set-matching and gang-matching are beyond the capabilities of this language.

3.1.3 Redline

The Redline system [LiFo03], developed at the University of Chicago, formalizes the constraint-matching process as a constraint satisfaction problem [Tsan93] and uses

constraint-solving algorithms to tackle the problem. As illustrated in Figure 2, Redline consists of a number of different components.

"The Redline language defines the syntax and basic semantics for the specification of descriptions. A description is a set of constraints describing either a request or a resource" [LiFo03]. Optionally a vocabulary can be used to define more expressive semantics. The vocabulary can be constructed utilizing an ontology language like OWL [W3C09] or DAML [DAML].The conflict-check engine examines if a set of constraints is consistent both with the Redline language and the vocabulary.

The logic used to match the request description to one or more resource restrictions is implemented in the matchmaking engine. Based on this basic functionality additional services, like an e-commerce searching engine or a grid resource selection service can be developed.

Redline allows the description of the resources to be different in generality and complexity [LiFo03]. Resources can be matched, even if for one resource every attribute is known exactly while for other resources only some general characteristics are known. In addition to matching based on properties Redline also can match resources based on policies, e.g., access policy. In contrast to the ClassAd language Redline is able to use both gang-matching and set-matching. Redline is still restricted by its origin in and focus on grid computing. Additionally, Redline so far is a pure theoretical work with no working implementation.

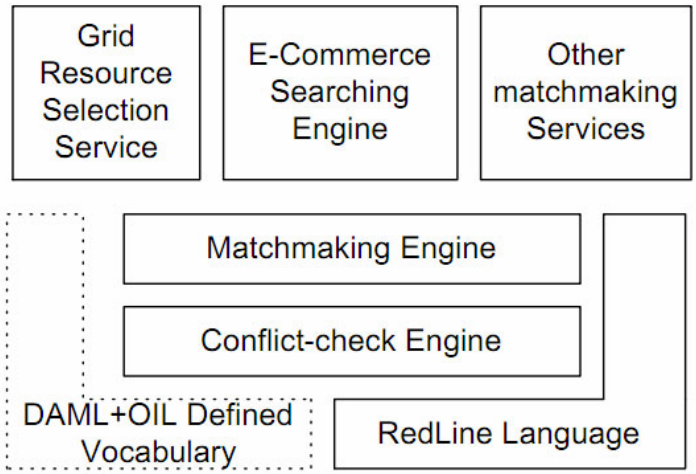


Figure 2: Redline Components, from [LiFo03]

3.1.4 Resource Description Framework

The Resource Description Framework (RDF) is a "standard model for data interchange on the Web" [RDF04]. It is a generic World Wide Web Consortium (W3C) recommendation which most often is implemented in XML. The RDF data model is represented through a set of statements about resources. Each statement is in the form of a subject-predicate-object expression [LaSw99] which is known as a triple. The subject signifies the resource, the predicate signifies a feature or a property of the resource and links the subject to the object which can be either another resource or a base data type. For example the sentence "An elephant is an animal" would result in a triple with the subject "elephant", the predicate "is a" and the object "animal". In the standard RDF, the code representing the triple would look like this:

```
<rdf:RDF>
 <rdf:Statement>
  <rdf:subject   rdf:resource="Elephant" />
  <rdf:predicate rdf:resource="onto;is a" />
  <rdf:object    rdf:resource="Animal" />
 </rdf:Statement>
</rdf:RDF>
```

RDF is a major part of the Semantic Web initiative [SemWeb] by the W3C. RDF itself is an abstract model, it is not tied to any specific data format and the triples can be represented in a number of ways. The most commonly used data formats are the XML serialization shown above and the N3 format [BeCo11] which prioritizes human-readability.

There exist several extensions of RDF to construct different ontologies. *RDF Schema* provides a limited number of elements to allow the design of simple ontologies. It introduces the class concept, class restriction of the subjects and objects and class hierarchies. It also establishes properties that describe a relation between resources and which also conform to a hierarchy.

The Web Ontology Language (OWL) [W3C09] which encompasses RDF Schema, allows a much broader array of ontologies. Since the development of the enhanced version of OWL 2 in 2008 OWL is no longer dependent on RDF as an exchange syntax but can be serialized using an independent XML serialization format [MPP09].

3.1.5 SPARQL

The SPARQL query language [PrSe08] is one of several languages that can be used to query RDF data and is a W3C recommendation since 2008. It was designed to be flexible enough to be useful in a wide variety of situations using a list of different use cases when describing the requirements [Clar05]. SPARQL uses a SQL-like syntax. A typical query contains a set of triple patterns called a basic graph pattern [PrSe08] which corresponds to the RDF triple of subject, predicate and object. The following query shows the basic structure of a SPARQL query, searching for a person by name.

The PREFIX allows the abbreviation of the URIs used to define the variables. The SELECT clause lists the attributes that are shown in the query result while the WHERE clause defines the requirements that have to be fulfilled so a match can occur. Additional constraints in terms of filters can be added to the WHERE clause to restrict the values of variables using regular expressions or to restrict arithmetic expressions.

Attributes which may be part of the result, but don't necessarily have to be, can be included by the use of OPTIONAL in the WHERE clause. For example not every person has a middle name, but by using OPTIONAL it is possible to add the middle name to the result set if the person has a middle name and leave it out if the person does not. The FROM clause defines the graph against which the query is matched. Utilizing the keyword NAMED it is possible to run a query using different RDF graphs at the same time as data source. Different queries in different RDF graphs can be added to the same result set. The keyword UNION can be used to define a query which can be matched by more than one graph pattern. If there is more than one match, all possible solutions will be displayed.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
FROM <persons.rdf>
WHERE
{
    ?person foaf:name ?name.
    ?person foaf:name "Johann Müller".
}
```

The result of a standard query is an unordered collection of solutions [PrSe08] and can be manipulated to make analysis and further processing easier. The ORDER BY clause arranges the results in sequence either ascending or descending. DISTINCT deletes duplicates while LIMIT allows only a specified number of solutions to be displayed.

In addition to the SELECT query shown above there exist three other SPARQL queries, CONSTRUCT, ASK and DESCRIBE [PrSe08]. While the standard SELECT returns the variables that have been bound in a query pattern match, CONSTRUCT builds a RDF graph out of these variables and returns the graph. ASK returns yes if the query matches and no if it does not. Finally DESCRIBE returns an RDF graph containing data about the resources.

While SPARQL is not the only query language that exists for RDF, it has emerged as the standard one. Its flexibility makes it perfect for a wide variety of fields of application and its relative simplicity combined with a similarity to SQL make it easy to learn.

3.2 Analysis of Existing Solutions

Combining process management with resource management is relatively new. Consequently, the number of both theoretical scientific works and implemented solutions is rather low. Nevertheless, there exist some interesting approaches.

Bussel [Buss98] criticized the user-initiated way in which workflow management systems are designed. "(T)his 'demand-based' initiation of workflow instances might cause tremendous problems for a company. Resource overload to meet a deadline or late deliveries due to limited resources can be the result of unconstraint workflow initiation. If there would be a way to schedule a workflow instance at its initiation time against the capacity of the required and available users as well as other resources, then resource overload can be avoided or late deliveries can be predicted" [Buss98]. He proposes a deeper integration of project management tools and workflow management systems. In general the integration should work on the execution logic level of both systems as shown in Figure 3 instead of the interface or the database level.

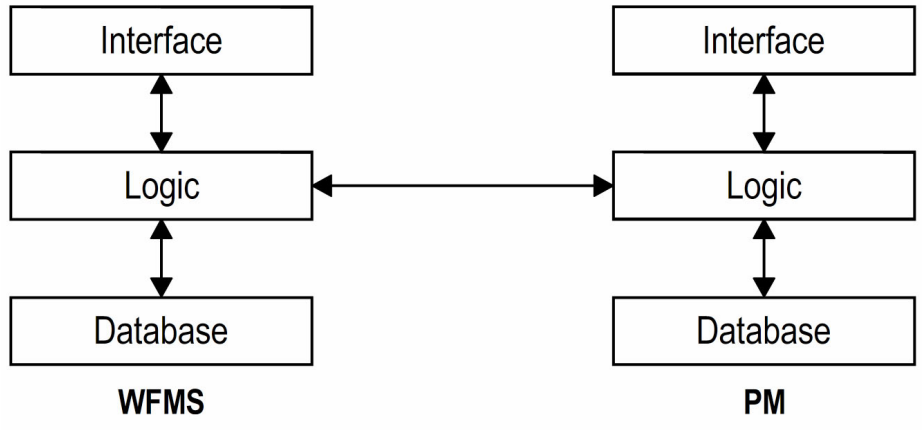


Figure 3: Integration of Workflow Management System and Project Management, from [Buss98]

The integration can be split up into two parts: schema integration and behavior integration. Schema integration takes the various concepts that make up both workflow management systems and project management and tries to map the corresponding concepts. The workflow tasks are mapped to project management tasks, workflow sequence is mapped to the end-to-begin relationship in project management etc. In cases where a direct mapping is not possible and the concept is not necessary for integration it simply is not used in the integration. If the concept is necessary it is introduced to the system or a workaround is used. The resource assignment is handled by the project management software, even if the workflow management system has some internal functionality for resource assignment.

Behavior integration ensures that the states of both the workflow management system and the project management software stay consistent with each other. If a state change in one system affects the other system it has to be defined which changes results from it. The behavior integration can be done in two different ways. The first approach lists all interfaces for the systems that need to be integrated. If not all interfaces can be integrated, "(t)he missing interfaces can be seen as requirements for one or both the systems" [Buss98].The second way of looking at behavior integration is detached from the particular systems. First an optimal integration is developed and from this the requirements for the two systems are derived.

Using behavior and schema integration it is possible to match any two project management and workflow management systems and arrive at the requirements needed to integrate the two. While the ideas put forth in this paper are very interesting-especially given the early date they were published-it is questionable if the integration of project management and workflow management systems is really the ideal solution. In most cases it seems more productive to enhance the functionality of workflow management systems by giving them the ability to organize and allocate resources themselves instead of having to work indirectly through project management software.

In contrast to the integration-based approach Russel et al [RHEA04] have assembled a list of workflow resource patterns to systemize the "various ways in which resources are represented and utilized in workflows" [RHEA04]. This list extends the work of the Workflow Patterns Initiative [WPI] that introduced a multitude of control flow and data patterns. While the paper mostly focuses on human resources the basic definition of resources and organizational structures allow for both human and non-human resources.

A resource is defined as "an entity that is capable of doing work" [RHEA04]. Resources are integrated into the structures which use them and can be part of an organization, or organizational units such as teams or groups or a division. They can have a specific position in the organization, subordinate resources for which they are responsible as well as roles and capabilities. Roles "serve as another grouping mechanism for human resources with similar job roles or responsibility levels" [RHEA04] while capabilities represent skills or other attributes that are important to their suitability for a specific kind of work. Every resource has a schedule and a history that represent which work items are scheduled or have been completed.

In addition to the resources Russel et al [RHEA04] define a couple of workflow-related terms. A *workflow* is "a description of a business process in sufficient detail that is able to be directly executed by a workflow management system" [RHEA04]. A workflow consists of a number of tasks which can be represented as a directed graph and an executed instance of a workflow is a case or process instance. The work, that a resource has to do, is assigned to a resource in the form of *work items*, each of which "describe an integral unit of work that the resource should undertake" [RHEA04]. Each work item has a lifecycle that starts when the work item is created and ends when it is either completed or fails. The lifecycle of a work item is shown in Figure 4. One can see how the work item is created, allocated to a resource, executed and-depending on the success of the execution-put in the completed or failed state.

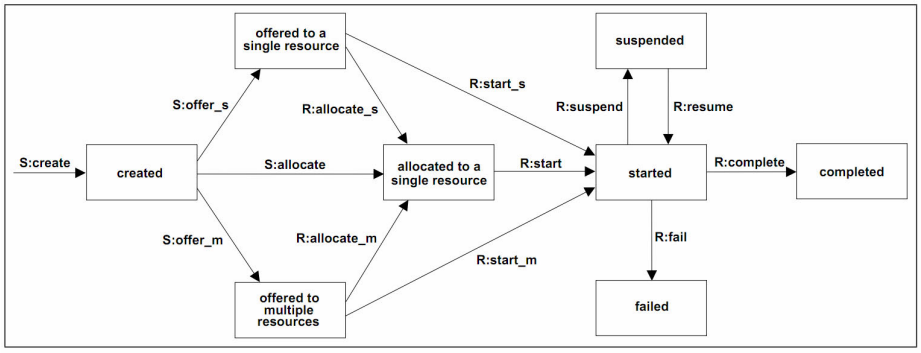


Figure 4: Work Item Lifecycle, from [RHEA04]

The patterns are at the heart of the paper. For every existing pattern a description, an example and related pattern are shown. Additionally the motivation for the pattern, its implementation in popular workflow engines, problems with this pattern (called issues) and their solutions are presented.

The patterns are grouped in a number of different types. Creation patterns govern how the allocation is handled at the creation of the resources which is represented by the S:create arrow in Figure 4. Push patterns represent the cases when the work item is offered to one or more resources or directly allocated to a single resource. This corresponds to the state transitions S:offer_s, S:offer_m, and S:allocate. Pull patterns describe that the resources have either seen the offered resource and decide to allocate or directly start the execution. The allocation is represented by the state transitions R:allocate_s and R:allocate_m and the start of the execution is represented by R:start, R:start_s, and R:start_m.

In addition to these patterns that govern the usual sequence of events, there exist detour patterns that deal with situations where the allocations that have already been made are interrupted. Examples include de-allocation of a work item, delegation and reallocation of a work item to a different resource or the repetition of the execution of a previously completed work item. Auto-start patterns represent situations where the execution of a work item is started automatically, usually skipping the R:start, R:start_s, and R:start_m state transitions of the pull patterns. Visibility patterns define if and how the availability of a work item can be seen by resources. Finally multiple resource patterns represent a small set of patterns that allow an interaction between more than one work item and one resource.

The work of Russel et al [RHEA04] represents a huge step forward in the understanding and systematization of resources in workflow systems. Like the similar work of van der Aalst et al with control-flow patterns [AHW03] it shows the possibilities of this technology and provides a benchmark for practical implementations that yield valuable information and guidance for further development. On the other hand, the work is also restricted in its scope. Apart from the limited multiple resource patterns Russel et al [RHEA04] deal only with a one-to-one

interaction between work items and resources. Additionally the focus is clearly on human resources which leaves non-human resources mostly left out.

Zur Muehlen [Mueh04] developed a set of core requirements that a resource model should satisfy "in order to be applicable to different workflow scenarios" [Mueh04]. He defines a resource as "an entity that is assigned to a workflow activity and is requested at run time to perform work in order to complete the objective of the activity" [Mueh04]. In contrast to Bussel [Buss98] he sees the management of resource information clearly in the domain of the workflow engine and defines four basic requirements that resource models in a workflow environment should fulfill.

The first of these requirements is robustness. A change in either the resource model or the workflow model should not change the other model in any way. Consequently, at least one abstract entity type is needed for the resource model to separate the resources linked by the resource model from those linked by the workflow model.

The next requirement, flexibility, stipulates that the resource model has to be flexible enough so the current organizational hierarchy can be moved to a new system without having to make compromises in the naming and interaction between the organizational units. It must be possible to change the types and names of entities and develope new relationships between existing and new entities.

The third requirement is scalability. The model has to account for new levels of hierarchy and make it possible to expand the organization. This applies especially in the case of mergers or internal restructuring of the organizational hierarchy.

Finally domain independence makes sure that the resource model is not fixed on a particular use case and does not make assumptions about the processes that do not hold in every case. The model has to be able to work with situations where only human actors interact with the process as well as fully automatic executions. Furthermore the model should only have a limited number of entity types to make it easier to maintain it.

Ouyang et al [OWF+10] developed a conceptual data model concentrating on the relationships between multiple resources that work on the same activity. The model is specially designed to fulfill a number of requirements [OWF+10].

The first requirement is *design-time resource specification and analysis*. The model has to include all information that is relevant for specification and analysis including what resources are involved and their respective quantities.

A further focus is *automated support for resource management*. Interaction between resources and tasks have to be supported by sufficient information. This includes resource assignment and delegation privileges.

The third requirement is *resource scheduling and availability*. The workflow system has to know about the current state of the resource but also has to take into account if the resource

is available in the future. Both a list of completed tasks and a list of planned tasks have to be included for every resource in the form of history and schedule.

Finally it must be possible to monitor the usage of all resources and collect the data necessary to analyze the process. For this, both start time and finish time of resource activities need to be logged as well as machine downtimes.

Based on these requirements Ouyang et al developed a resource conceptual model using the Object Role Modeling notation [Halp01]. The model is divided into four different modules: resource classification, multiple resources assignment, resource calendar and resource utilization logging.

Resource classification defines how the resources are structured. The main difference made here is between human and non-human resources. Human resources hold a position within an organization and possess roles and capabilities that define which tasks can be transferred to them. They also can be part of a team, which is an organizational group that is not part of the normal hierarchical structure of the organization.

Non-human resources are divided into two different types, application and non-application resources. Application resources are software-based resources like web services which are identified by a URI. Non-application resources can be machines or equipment which is classified as durable. Alternatively they can be raw materials or input products which are defined as consumable. Application resources usually have a capacity which can mean different things depending on the type of the resource. For example "a hospital may have seven operating rooms, a sound tape may provide up to an hour of recording, and an oil container may hold up to ten liters of engine oil. For consumable resources (...), their capacity can be used to estimate their remaining useful life, while for durable resources (...) their capacity may indicate the duration of their maintenance cycle" [OWF+10].

The second module of the conceptual model is *multiple resources assignment*. As shown in Figure 5, a task is managed by a primary resource which can either be a human resource in the case of a user task or a application resource in the case of a system task that does not need human intervention. In addition to these primary resources a user task can require additional resources regardless of their type which can be either addressed directly or by their role or usage type.

The third component of the model is the *resource calendar* which contains information about the status, location and business rules of a resource. The availability status is presented as one of a fixed number of states, e.g., available, occupied, blocked, etc. Every status has a time period associated with it and can also have a reason why the resource has that particular state.

Similarly, a resource can be at a specific location for a certain time period. For every resource there can be a number of business rules which fall under four different categories. State transition rules govern under which conditions the state of a resource can change from beginning state to end state. Work-interval rule and work-limit rule show the legal or

organizational mandated limit for regular breaks and maximum working hours. The multi-tasking rule defines how many different tasks a resource can carry out.

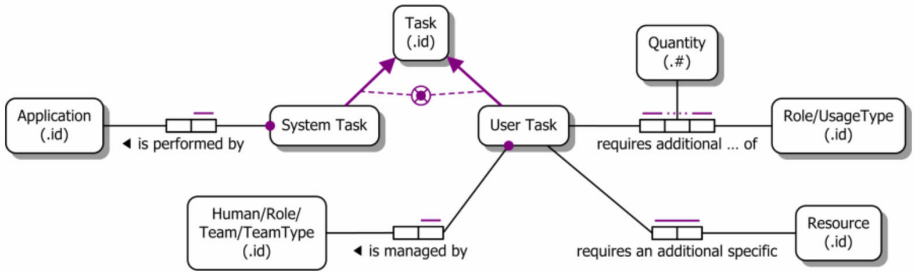


Figure 5: Multiple Resources Assignment, from [OWF+10]

Finally, *resource utilization logging* shown in Figure 6 provides the capture of information about the process execution using resources. A work item is the instance of a process task which uses a certain quantity of a resource at a certain location. Additionally both the start time and the end time are logged with the resource utilization.

The work of Ouyang et al provides valuable insights into the working and interaction of multiple resources. Unfortunately, the resource model they put forward is fixed in both the categorization of resources and in the properties the various resource types. No further user-driven customization is possible and the arbitrary use of capacity for a number of completely different things shows the need for a greater number of user-defined features. Nevertheless Ouyang et al introduce a great number of interesting ideas and requirements that help to build a basis for any system that works with multiple resources.

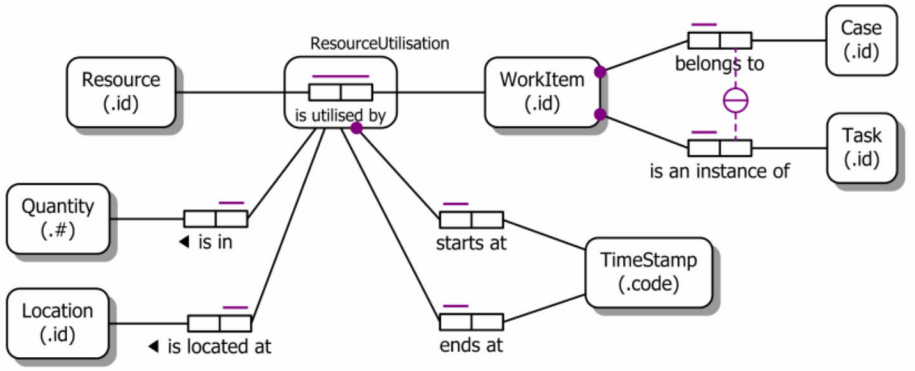


Figure 6: Resource Utilisation Logging, from [OWF+10]

4 Resource and Process Modeling

Of the three components process modeling, process execution and process monitoring, the modeling phase is the one where the greatest interaction between resource management and process management takes place. In this phase the main decisions have to be made regarding the structure of the resource model as well as the changes that have to be made to the process modeling. This chapter is divided into two different parts. Chapter 4.1 takes a deeper look at the resource modeling, establishing the requirements in 4.1.1, touching on resource specification and classification in 4.1.2 and 4.1.3 and defining the resource features in chapter 4.1.4.

Chapter 4.2 examines the issues that arise from integrating resource modeling and process modeling. The requirements of this integration are discussed in chapter 4.2.1, the structure of resource requests in 4.2.2, scopes of process steps in 4.2.3, process rules in 4.2.4 and resource constraints in chapter 4.2.5.

4.1 Resource Modeling

The resource model itself is one of the main components for the integration of the resource perspective with process modeling. This chapter deals with the way a resource model has to be designed to fulfill all the general requirements discussed in chapter 2.3. Following the definition of zur Mühlen [Mueh04], in the context of this work a resource is defined as "an entity that is assigned to a workflow activity and is requested at run time to perform work in order to complete the objective of the activity".

4.1.1 Requirements Resource Modeling

Based on the general requirements for the complete system put forth in chapter 2.3 there result concrete requirements for the resource model itself.

1. Accuracy: A model is always an abstraction of reality and therefore can never map the complexities of the real world perfectly. Nevertheless the resource model has to be an accurate and correct interpretation of the real resources that it aims to represent. Deciding in which cases a specific detail in the resource model is needed can be tricky and it is not immediately apparent which details can be dropped and which are essential for the model.

It is important to remember that the process model will be the main interaction point for both users and modelers of the process management system between them and the real resources. Every way in which the resource model behaves differently than the real resources has the potential to introduce (sometimes catastrophic) errors into the work operations of the organization.

2. Simplicity: Never model more than is necessary for the job. Each additional level of complexity makes it much harder for the modeler to design and for the user to understand resource composition and interaction. For a resource model to work it has to represent the complexities of the real resources in a way that makes it easy for the user to understand.

3. Coherent classification: The resources have to be classified in a way that takes into account their inherent differences and the variety of resources. The biggest difference between resources is the difference between *human* and *non-human resources*. While it is desirable to treat them the same way to keep the model as simple as possible the types are just too dissimilar to make this a real possibility. The positions that human resources occupy in an organization are fundamentally different from the positions non-human resources occupy.

   Human resources hold certain positions in an organization and they fulfill specific roles. This has to be represented in the model. Human resources also have certain skills and capabilities that have to be modeled since they often can be the basis for deciding which human resource is used for a certain task.

   Non-human resources on the other hand do not have the special positions, roles and capabilities that human resources have. Non-human resources are so diverse, however, that they have to be divided into different types, so that raw materials and locations can be treated differently than machines and tools.

4. Design flexibility: In addition to the hardwired features that result from the coherent classification requirement it has to be possible to extend the resource entities with all the features that are required to represent them accurately. This involves an unlimited number of attributes with a wide array of different data types. Additionally there has to be the possibility to model relationships between different resources. The modeler must also be able to define types of resources that share certain similarities and provide them with a common set of attributes and relationships.

5. Resource hierarchy: It must be possible to specify connections between resources to account for their inherent hierarchy. Some resources are subtypes of other resources, some share certain features, but not others. To support this in the resource model resources must be able to inherit attributes, capabilities, roles and positions from other resources. The capabilities, roles and positions themselves must also be able to inherit their structure from other capabilities, roles and positions. A capability "can use Open Office" for example should be a subtype of the capability "can use word processor".

It is apparent that these different requirements influence each other and at times even represent a conflict as depicted in Figure 7.

Especially the goals of accuracy and simplicity often are at odds with each other. Additionally, both influence the other requirements.

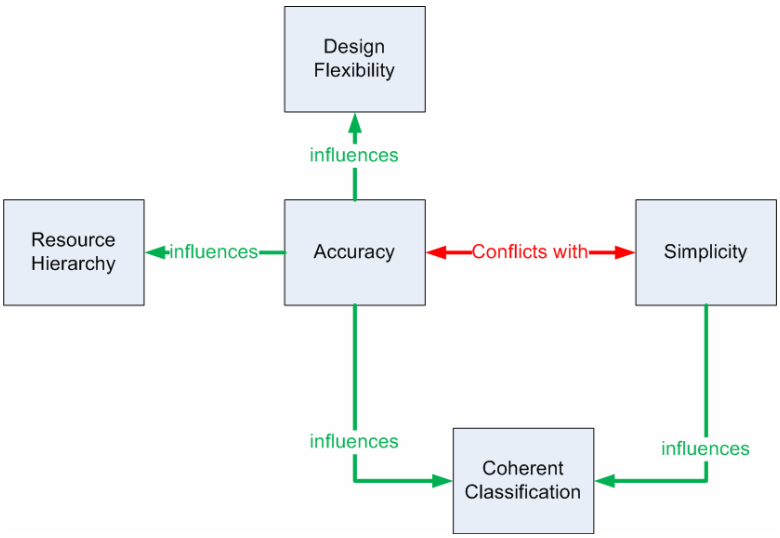


Figure 7: Requirements Resource Modeling

4.1.2 Resource Specification

The specification of the resource model is independent from the process view and consists of a number of different concepts.

There are two different viewpoints on resources. The first concerns itself with *resource classes*, the abstract concepts that represent a specific type of resource. The second viewpoint concentrates on *resource objects*, the actual representation of the physical resources that are used in the production context. This dualism is the result of the requirement to give the modeler the freedom to design any type of resources for himself if he so chooses.

Every resource class has a name, a *parent resource class* and a list *of property classes* that belong to the resource class. The name is unique, so the resource class can be identified unambiguously. The parent of the resource class defines from which superclass the resource class inherits its properties. If the resource class does not have a user-defined

resource as parent, its parent is the basic resource. The basic resource has the name "Resource", no parent and no property classes.

Property classes have a name and a *data type*. They exist in two different types, *attribute classes* and *relation classes*. Attribute classes represent either single objects with a data type or they can be lists of objects with the same data type. Relation classes in turn signify a connection between two resources. Their data type can only be an already defined resource class or a list of resource classes. Resources can inherit all types of attribute classes and relation classes from other resources. Figure 8 shows the basic structure of resource classes with their accompanying properties.

Example 1 presents a set of resource classes with their property classes.

The second viewpoint on the resources is that of *resource objects*. These are representations of the actual resources that an organization has at its disposal. They consist of a unique name, the resource class which they belong to and a list of property objects and the resource calendar.

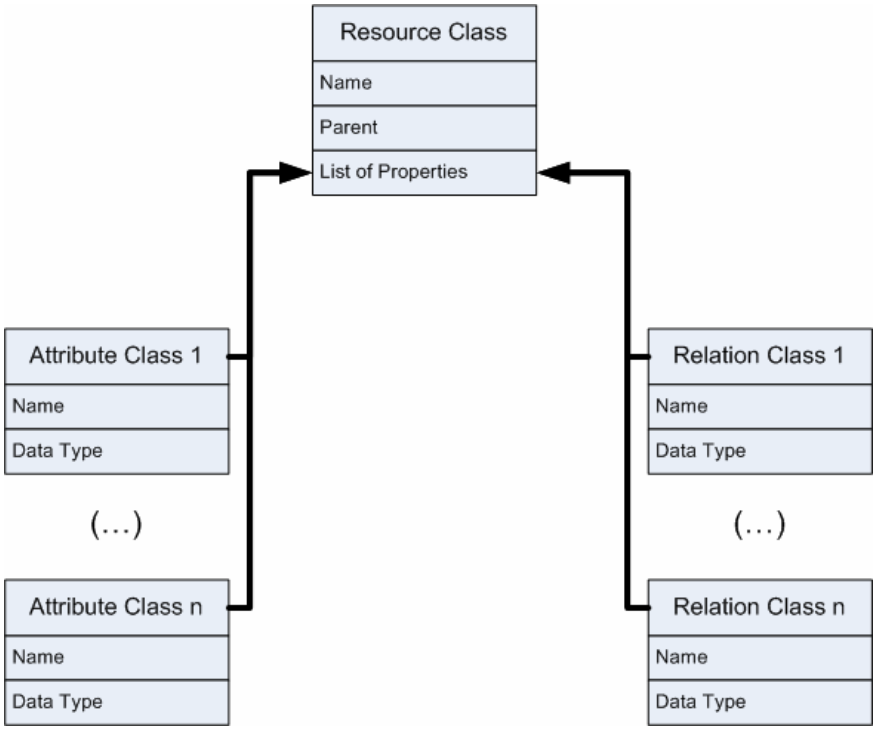


Figure 8: Resource Class Structure

Example 1 (Resource Classes)

This example describes the structure of a set of resource classes.

All sets of resource classes start with the basic resource, named "Resource". "Resource" serves as a parent for all the other resources. It has neither attributes nor relations and cannot be instantiated. It serves only as the blueprint and anchor for the other resources. Two resource classes inherit directly from "Resource", "Motor vehicle" and "Trailer".

"Motor vehicle" has no inherited properties, but three user defined attributes. They are ID, peed in km/h and weight in kg. All attributes have INTEGER as data type. "Trailer" has also no inherited properties, but three user defined attributes. They are ID, weight in kg and maximum load in kg. Neither "Motor vehicle" nor "Trailer" have any relations.

"Motor Vehicle" has two resource classes as children, "Car" and "Truck". Both have the inherited properties ID, speed and weight. "Car" has an additional attribute, color with the data type STRING. "Truck" also has an additional attribute, maximum load with the data type INTEGER. "Truck" also has a relation, attached trailer. The data type of this relation is "Trailer". "Car" does not have any relations and no resource class has any further children.

The set of resource classes is presented in Figure 9. Inherited properties are shown in red, while user-defined properties are shown in black. Parentage is shown by a black arrow, the data type of relations is shown by a blue arrow.

Every *property object* has a value and link to the property class it belongs to. The value of the property object has to adhere to the data type of the property class. As with the property classes property objects are divided into *attribute objects* and *relation objects*. In that way the resource class defines what type of property objects can exist for a resource object.

The list of resource objects that belong to a system is called a resource pool. This signifies all the resource that are potentially available for use by the business process management system.

Example 2 shows two resource object with their properties.

Example 2 (Resource Objects)

This example shows the structure of two resource objects with their properties.

There are two resource objects, a truck and a trailer. The unique name of the truck is "Truck No. 74312", the unique name of the trailer "Trailer No. 6574". The resource class of "Truck No. 74312" is "Truck" and the resource class of "Trailer No. 6573" is "Trailer".

The truck has a total of four attribute objects and a relation object, the trailer has three attribute objects. The attribute objects don't have a name or ID and so can only be identified by their resource object. The attribute object of the truck show that he has an ID of 74312, a speed of 140 km/h, a weight of 7.400kg, and a maximum load of 11.600 kg. The attribute objects of the trailer show that he has an ID of 6574, a weight of 500kg and a maximum load of 2.500kg. The relation object of the truck belongs to the relation class attached trailer which as seen in Example 1 has a data type of "Trailer". The value of the relation is the resource object "Trailer No. 6574".

A graphical representation of the two resource objects and their properties can be seen in Figure 10. The attributes objects belonging to a resource objects are linked by a black arrow, while the relation objects are linked by a blue arrow.

4.1 Resource Modeling

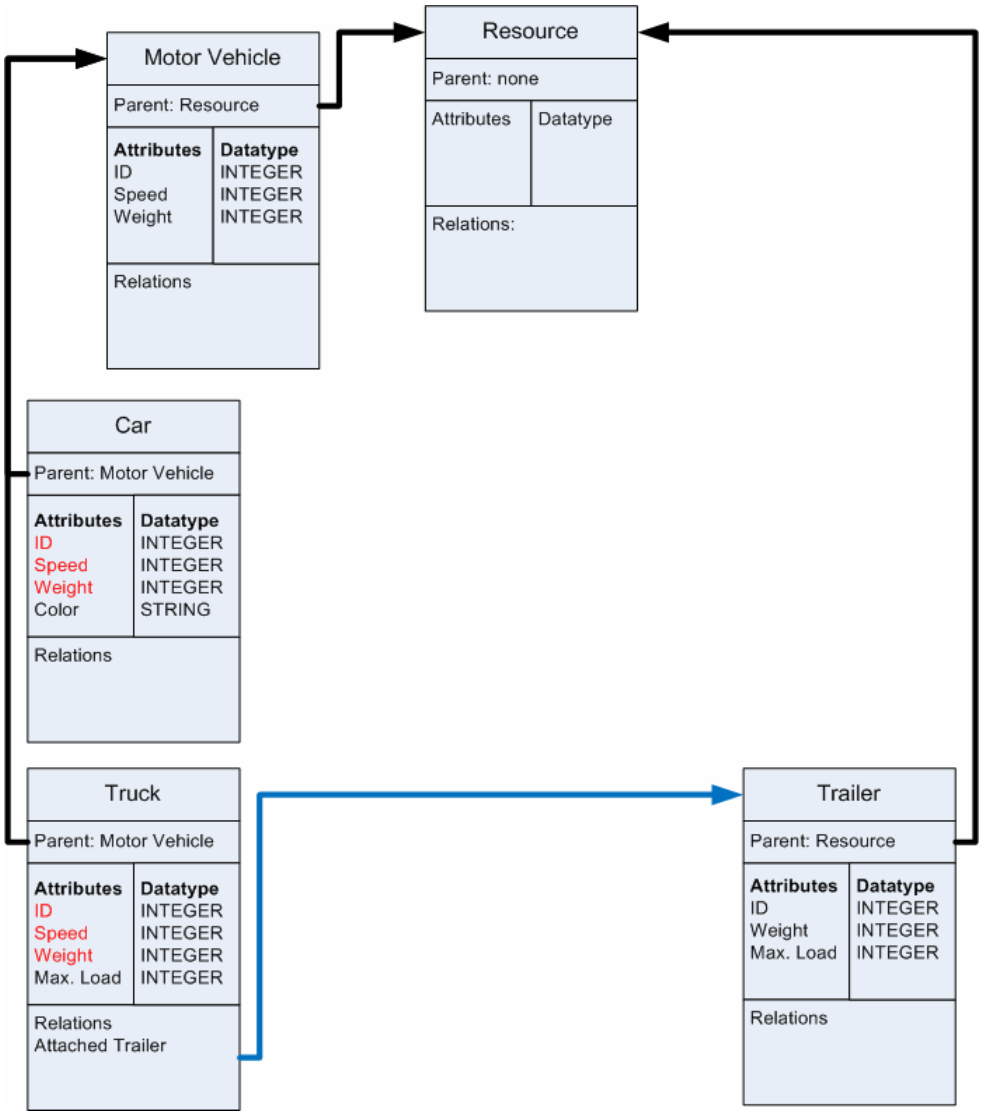


Figure 9: Set of Resource Classes

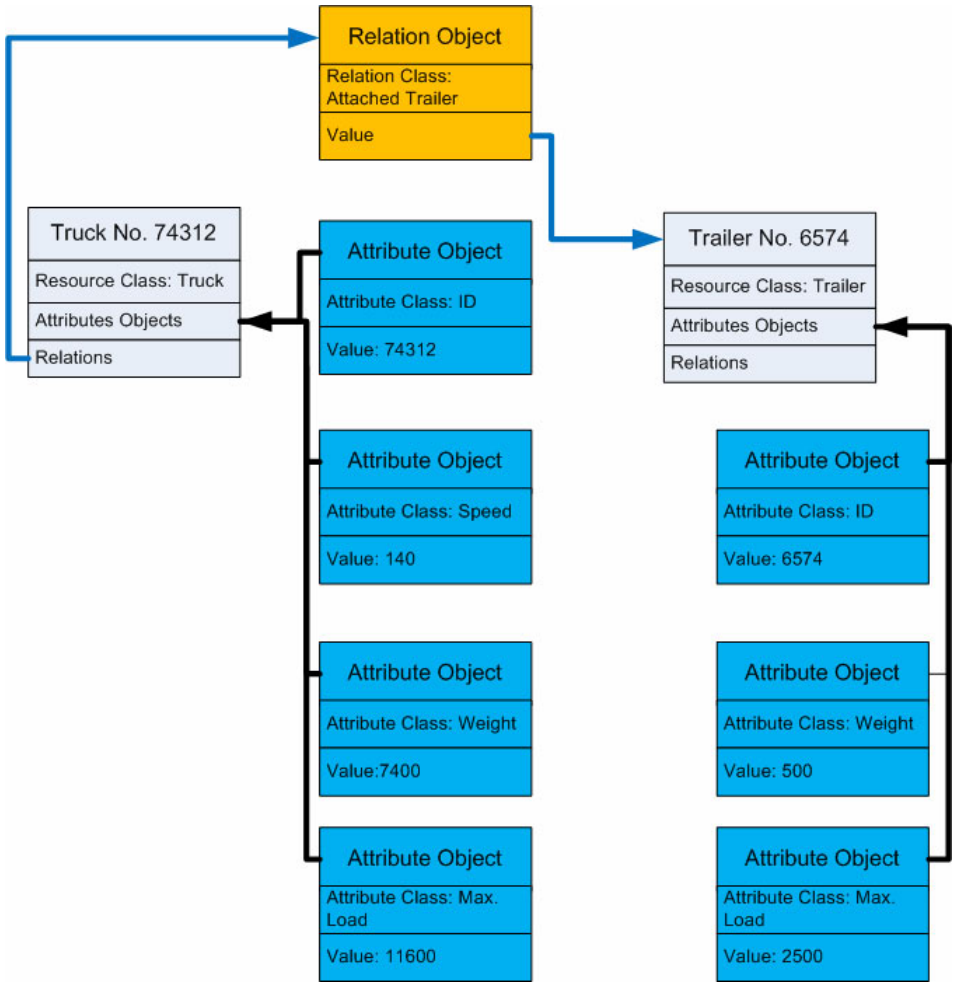


Figure 10: Example Resource Objects

4.1.3 Resource Classification

Resources can be subdivided into three different groups which can be seen in Figure 11. Depending on which group a resource belongs to, it may have different properties and some operations can only be performed on specific resource groups.

The first of these groups are *human resources*. Human resources are not divided further. The user defined features and hierarchies enable the modeler to specify the various types of human resources.

4.1 Resource Modeling

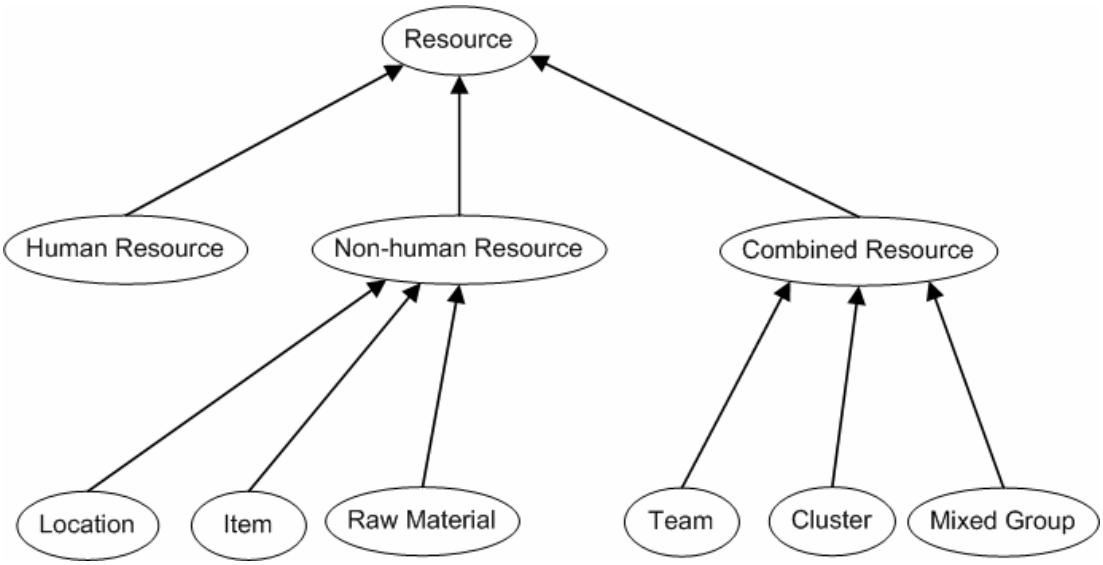


Figure 11: Resource Classification

Resources that don't represent a person are defined as *non-human resources*. They can be anything from a building, a production line or a type of office supply. The standard type of non-human resource is the *item*. This includes most non-human resources like machinery, work equipment or even virtual resources like a web service. There are two special types of non-human resources, *raw materials* and *locations*. Raw materials are handled differently than normal non-human resources, because they don't get used and reserved as concrete entities but represent a specific quantity of consumable resources. They have three special properties that are unique to raw materials, quantity, reserved and allocated. Quantity stands for the amount of the raw material that is available, reserved is the amount that is reserved by an activity to be used in the future and allocated is the amount of the material that is already allocated to a process before it is used. Locations are mostly used to link a number of resources together as being in the same place, so a factory does not use resources that belong to another factory thousands of miles away. They have coordinates that indicate where they are located and are linked to other locations by a distance attribute. This can either be the literal distance between the locations, but may also be used to specify the cost of transferring resources from one location to another.

The third group are the *combined resources* or resource groups. They represent a combination of different resources. If all the resources that are part of a combined resource are human resources, the resulting resource group is called a *team*. If the resources that are combined are non-human resources the resulting resource group is called a *cluster*. Combined resources that are comprised of both human and non-human resources are called *mixed groups*. Combined resources exist mainly to make the work with predefined resource

groups easier. In a hospital setting for example there are sets of medical objects that are required for certain treatments. The composition of these sets hardly ever changes. Consequently it would not be practical to model each individual component of the set and make them together part of a complex query when it is possible to define them as a cluster and require them as a single resource. In much the same way departments or subdivisions that are not fully integrated in the workflow and resource management system can be defined as a team that is required for a certain task. The division of labor within the department then can be managed informally instead of requiring information about all team members and their respective skills and capabilities.

In Figure 12 the composition of combined resources is shown. The arrows signify that an object is a special type of the linked object, so are, e.g., human resources a special type of resource. The divided box indicates a relationship between two objects, in the way that, e.g., a team consists of human resources or-seen the other way-a human resource can be a member of a team.

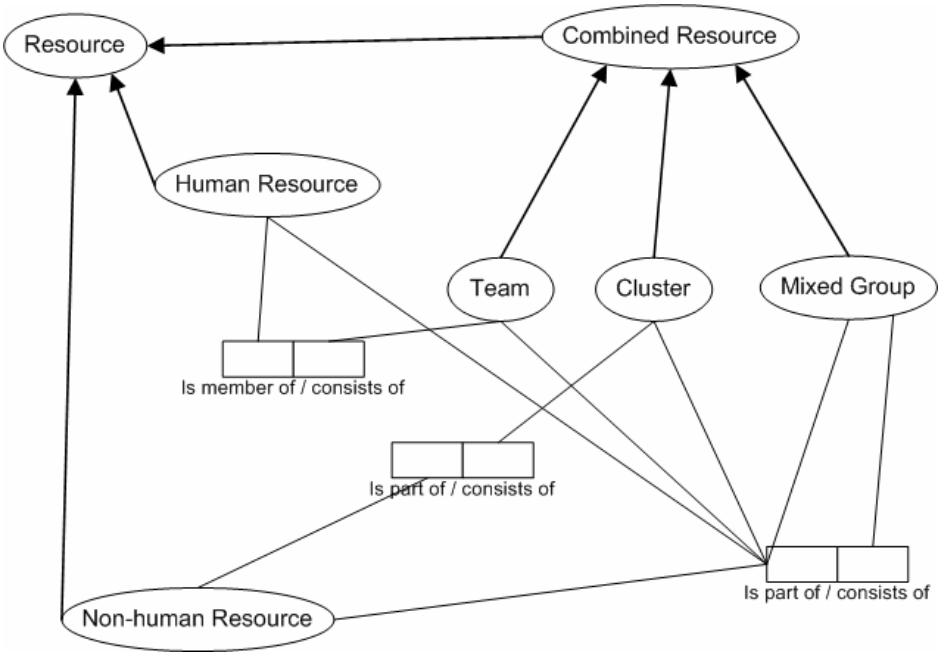


Figure 12, Combined Resources

4.1.4 Universal Resource Features

As discussed in chapter 4.1.2 resource object can have multiple user-defined properties. Additionally, they have a number of preset features. These features are availability state, global substitute, and resource calendar. These universal features are shown in Figure 13.

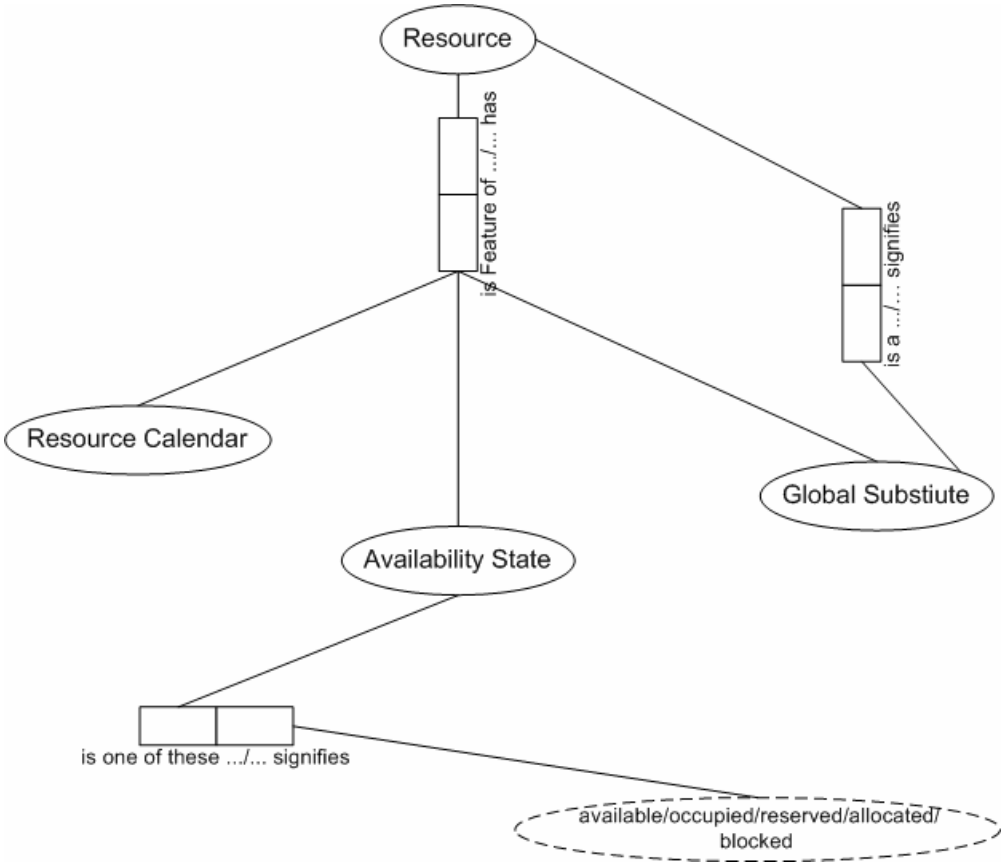


Figure 13, Universal Resource Features

All resources have an availability state. This feature is used as an indication in what situation the resource currently is. The availability state shows to the system, if the resource is in use by an activity, if it is blocked due to maintenance or management decision or if it is free to use. The different states that a resource can be in are described in Table 1.

The standard sequence of availability states that a resource object passes through is shown in Figure 14. First the resource object is in the standard state, it is *available*. Then the

resource object gets *reserved* by an activity. When the execution of the activity, that needs the resource, draws near, the resource is *allocated*. At the time of execution the resource object is put in the *occupied* state. After the process step is concluded the resource object is made *available* again.

This turn of events is of course not inevitable. A resource object can be allocated without being reserved first, an object can pass directly from available to occupied and of course the object can be blocked regardless of which availability state it inhabits at the moment. In theory a resource object can change from every resource

Example 3 shows a more complex transition between availability states for a resource object.

available	The resource is in working order and is neither used by nor intended to be used in the near future by any activity. The resource is available for use by any activity that has the right to access the resource. This is the default state.
occupied	The resource is in use by an activity.
reserved	An activity is planning to use the resource in the future and therefore has reserved it.
allocated	The resource is allocated to an activity before it is used.
blocked	The resource is neither in use nor intended to be used in the near future by any activity, but nevertheless is not available. This can be due to defects, management decisions or other reasons.
unavailable	The resource is not available for use. This is an umbrella term for the states occupied, reserved, allocated and blocked.

Table 1, Availability States

Resources also can have a *global substitute*. This represents a resource that can be used instead of the resource itself if the resource is unavailable. Substitutes are common for human resources where employees can be unavailable for longer period of times because of vacations or medical conditions. Every resource also has a *resource calendar*. In this all changes that affect the resource are saved together with a timestamp of when the change happened.

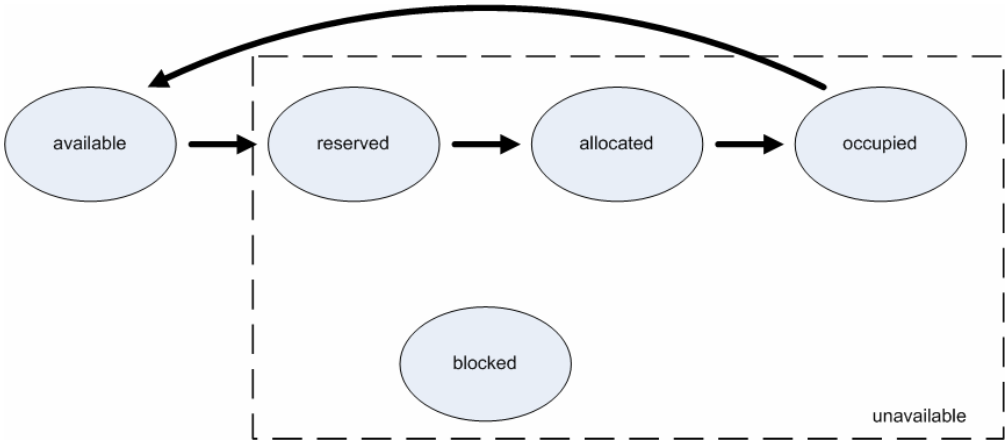


Figure 14: Standard Transition of Availability State

Example 3 (Transitions between Availability States)

This example describes how the sequence of availability states for a resource object can differ from the standard transition.

As usual, the resource object starts out as *available*. Then the resource object is *reserved* by an activity that is supposed to take place in the near future in **step 1**. Unfortunately the activity is stopped before it gets to be executed. The resources therefore returns to the *available* state in **step 2**. In **step 3** the resource object is *reserved* by a different activity. After a short waiting time the resource object is then *allocated* in **step 4**. When the activity is executed, the resource object's availability state is changed from *allocated* to *occupied* in **step 5**. A malfunction of the resource object occurs and it is transferred to the *blocked* state for maintenance in **step 6**.

This sequence of events is shown in Figure 15.

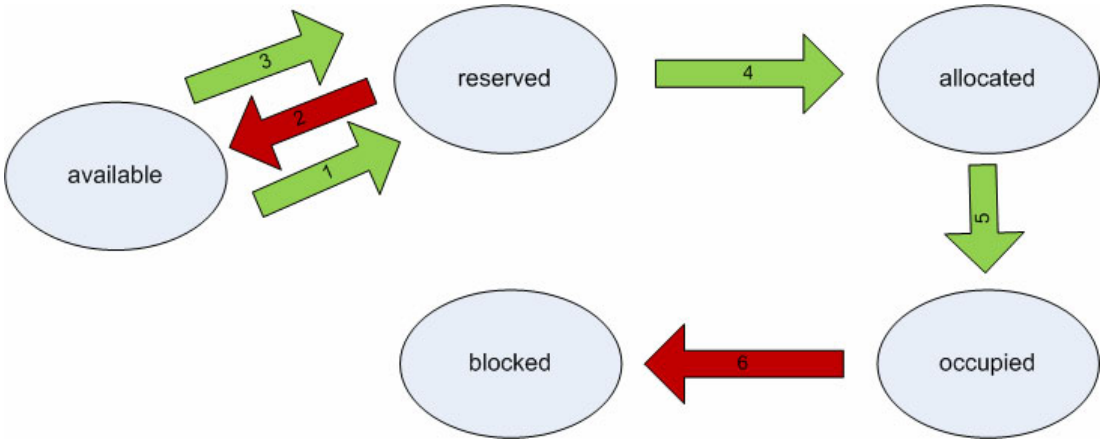


Figure 15: Example of Transitions between Availability States

4.1.5 Human Resource Features

In addition to the universal resource features described in the previous chapter, human resources have some extra features that make only sense for human resources.

Capabilities allow the resource designer to specify particular abilities or skills of a human resource. This can be the ability to use a certain machine, the skill to work independently on a problem or a certification of a degree that is needed for a specific task. Capabilities are especially suited for queries since the aptitude of a person for a task often depends on one or several abilities or skills. Capabilities are different from properties because they do not represent a specific attribute or relation, but explicitly state which skill or ability a human resource has.

The second feature that is relevant for human resources is their *position* within an organization. This is the formal position that the person inhabits within the organization. Usually a person only has one position. A position always belongs to an organizational group which can be a department, a branch or even the complete company depending on the organizational structure.

Of course the formal position is not enough to represent the various structural and hierarchical relationships that exist for human resources in a large organization. Therefore a human resource also has *roles* that display all the functions that are not defined by the position itself. This can be project specific like project leader or permanent like equal opportunity officer or data protection officer. These are also often used in queries to select a special type of human resource.

These special features that only exist for human resources are shown in Figure 16.

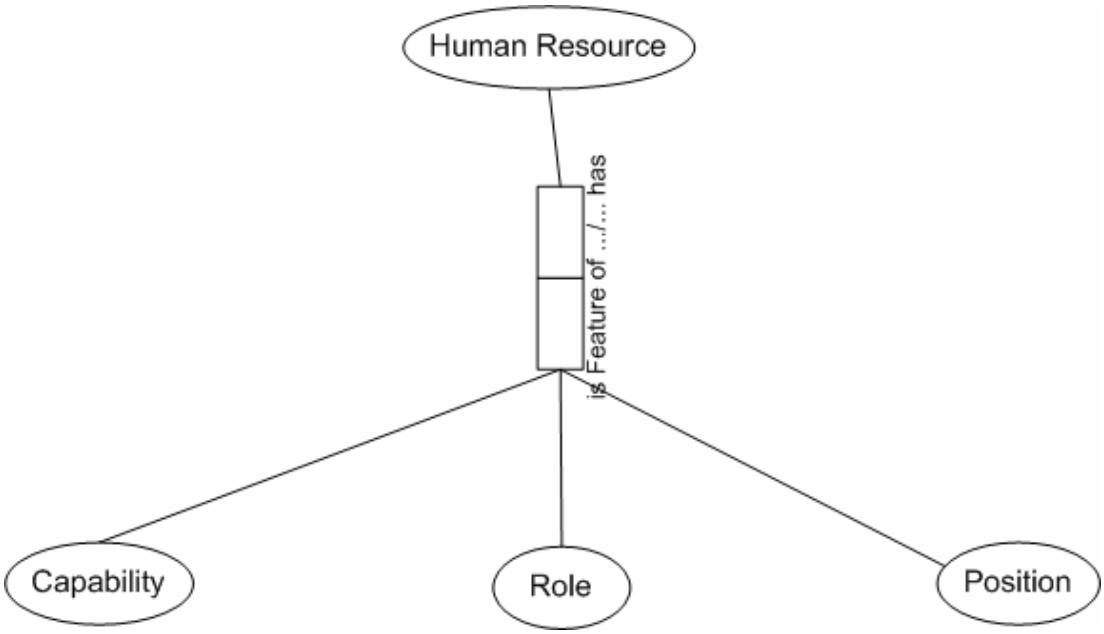


Figure 16, Human Resources Features

4.2 Integrating the Resource Model with the Process Model

The resource model developed in the previous chapter is relatively self-contained and can be used independently from any business process management system. This chapter deals with the requirements and changes that have to be applied to design a functioning system that integrates the resource model and the process model.

4.2.1 Requirements for Integrating the Resource Model with the Process Model

The requirements that follow for the process model are based on the general requirements that are identified in chapter 2.3.

1. Flexible resource assignment: The process model has to be able to incorporate a number of ways in which resources are assigned to an activity. The assignment shouldn't directly link the activity to an existing resource object but rather define a query by which the resource will be matched later. Obviously, this can be used to specify a single resource by a using unique ID, but there exist many more possibilities.

Resources can be selected by using their type, their position in the organization, their capabilities or their readiness to work together with other resources. The process model must allow all these different ways without overwhelming the modeler.

The decision which language should be used for queries has a significant impact on the way the system will behave. The more flexible and powerful the language is, the more powerful the integration of resource and process model will be. On the other hand, the query language has to be understandable by the modeler. Which query language should be used is obviously also influenced by the field of application. If the process modeling system is limited to a field where the restricting resources are mainly clusters of computers, e.g., scientific workflows, then a language like ClassAd or Redline would probably match the requirements best. If the aim is to integrate resource and process management in a way that it can be used in a wide variety of domains and with many different types of human and non-human resources then a flexible and open data structure like RDF in combination with a query language like SPARQL should be considered.

2. Multiple assignment and assignment hierarchy: In the real world it is common that more than one resource is needed to execute an activity. Therefore, the process model has to provide the possibility to assign more than one resource to an activity at the same time. In the case of multiple resources it has to be made clear which of the resources is responsible for the successful execution of the process step. Consequently, one of the resources has to be declared as primary resource to indicate which resource is the "owner" of the activity.

3. Resource-based constraints: In addition to any other constraints the process management system enables, constraints between activities based on resources are required. This is necessary to make sure that the resource object of a later activity can be selected depending on which resource objects have been used in previous activities. For example a financial company might want to have a different person auditing a transaction than the one who authorized it. Similarly, a customer service company might want to match the same operator to a customer for every activity during a longer service process if possible.

   A differentiation between constraints should also be possible. For example, some constraints may be more important than others. Constraints that result from legal requirements, for example, should be classified as hard constraints and may not be circumvented. Other constraints are not as critical and may be ignored if that means that the process can be continued instead of having to wait for a resource to become available again.

4. Flexible transition between process and activity: It is important that the process enables the designer to specify a section that comprises more than a single activity, but doesn't encompass the complete process. Many resource restriction and rules concerning the selection of resources rely on more than one activity but do not hold true for the complete process. Without the possibility to define certain ranges all task

would have to be chopped up into subtask to make the restrictions possible which would severely complicate the work of the process modeler.

5. Fine-grained Control of Resource Behavior: It should be possible to control the allocation and other behavior of the resources not only from a process or activity viewpoint, but a viewpoint that takes into account a small number of activities. Automated decision must sometimes be made for a couple of activities that are linked together. The system has to provide a tool to group these activities together. This includes, e.g., keeping resource objects reserved between the execution of two activities, because the activities are so closely linked together.

As shown in Figure 17, the different requirements directly influence each other. A flexible resource assignment is the precondition for both the flexible transition between process and activity and the multiple assignment and assignment hierarchy. In a similar way resource-based constraints and a fine-grained control of resource behavior are mutually dependent.

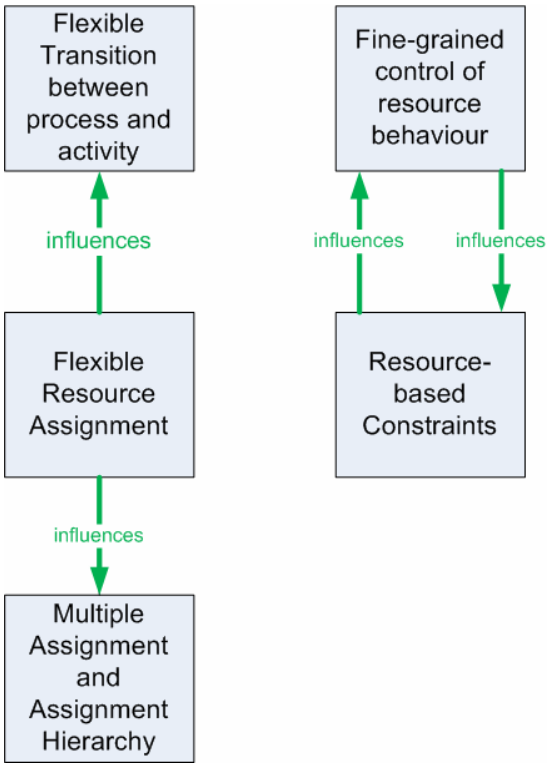


Figure 17: Requirements Process Modeling

4.2.2 Resource Requests

The main way in which resource management and process management interact during the process modeling phase is the specification of the resource needed for the execution of an activity. This is handled by using *resource requests*. For every activity it is possible to describe which resources have to be present so it can be executed. Every activity needs at least one resource object that executes the activity. The complete resource usage of an activity is condensed into one resource request, so there exists exactly one resource request for every activity.

A resource request consists of two different component. The first component is a list of all the resource classes that are needed for the activity. Every resource class is identified by a variable name.

One of the resource classes has to be classified as the *primary resource*. The question which resource will be designated as primary is of premier importance. This is the resource which will execute the activity and will be responsible for its success or failure. It can decide about the outcome of the activity and reserve or allocate other resource objects for future activities. If the request consists of both human and non-human resources the primary resource has to be a human one. An activity can have as many additional resources as the modeler decides and there exists no further differentiation between the non-primary resources.

Apart from the fact that primary resource executes the activity, the meaning of primary resource in the actual work situation may vary. In the event of a machine operator who is the primary resource with his machine the additional resource it is clear that all the decisions and responsibilities lie with the primary resource. The additional resource is only a tool used by the primary resource and has no further impact on the execution of the activity.

By contrast consider a mixed work group drawn from different departments and consisting of a number of specialists with a defined area of expertise. In this case the primary resource will only be responsible for the organizational work, making sure that the team works together well enough to accomplish the task. It doesn't necessarily mean that the primary resource makes all the decisions on how the activity is conducted.

The second component of the request is the *query*. Due to its great flexibility and relative simplicity SPARQL is used as query language. The structure of a resource request is shown in Figure 18. Every resource request also has an unique ID by which it can be identified.

The essential structure of the SPARQL query used in the resource request is shown below. The SELECT clause defines the properties of the resource which are supposed to be displayed. This is automatically generated by the system, so the ID of the selected resource object will be displayed. The FROM clause specifies the RDF graph which is queried. This represents the selected resource pool and is also automatically picked by the system. In the WHERE clause the variables given in the list of used resources are translated into SPARQL statements.

```
SELECT
FROM
WHERE { FILTER }
```

This automatically generated skeleton provides the background for the user input. The user simply decides by means of which properties he wants to restrict the resources and inputs his restrictions in the FILTER part. This can range from simply specifying the unique IDs of the resource objects he wants to building complex dependencies of the different resource objects of the resource request. By using placeholder variables SPARQL can even include resource objects into the query which belong to a different resource request and a different activity. It is relatively easy to build a graphical user interface that automatically generates the FILTER part of the query out of the user selections. This allows even users with no knowledge of SPARQL to easily and efficiently generate complex resource requests.

Example 4 shows a resource request with the list of used resources and query.

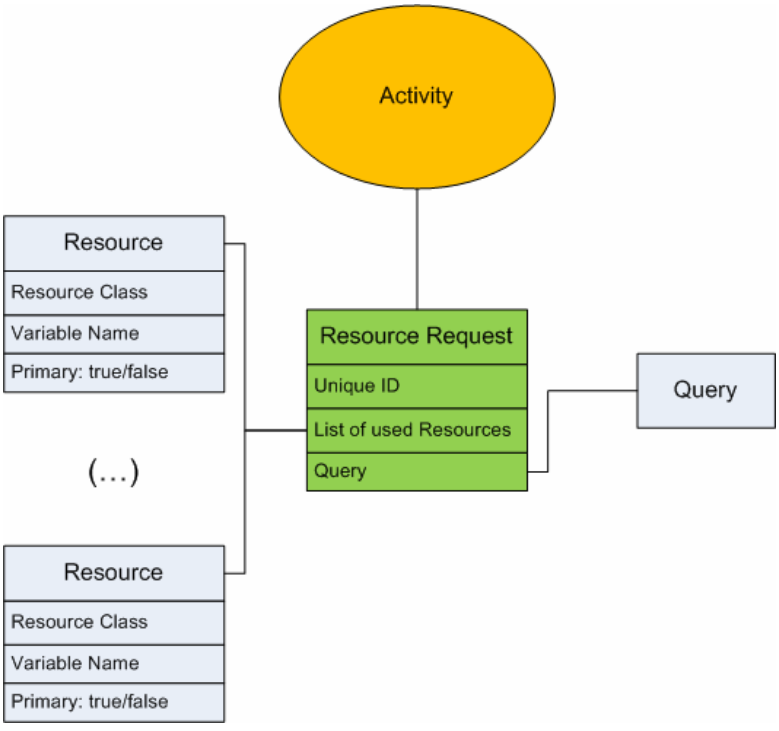


Figure 18: Resource Request Structure

Example 4 (Resource request)

This example describes a resource request that searches for a combination of truck and trailer that can work together.

First a primary resource is selected. In our case the resource class for this resource is "Truck". Note that while in most cases the primary resource is a human resource, this is not necessary and the primary resource can be a non-human resource like a truck. The variable name that is given for the truck in the query is "truckX". As an additional resource the resource class "Trailer" is selected. The variable name for that resource is "trailerY".

Out of these two resource classes the skeleton for the query is generated. In the SELECT clause the ID for the truck and the trailer is selected to be displayed. The resource pool for this query is located in the file "motorpool.rdf" which is inserted into the FROM clause. From the two resources the system automatically generates the SPARQL statements that define the class of the resources in RDF terms and define that the ID displayed in the SELECT clause belongs to the two resources.

Then the user can input the restrictions he wants for the resources into the FILTER clause. In our case we want a truck with a speed of more than 120 km/h. Additionally the maximum load of the truck should be more than the weight of the truck and the weight of the trailer combined.

After this input the system generates the SPARQL statements in the WHERE clause that are needed to link the specified restrictions to the resources and the query is complete. The ID is automatically generated as 3765. Figure 19 shows the completed resource request.

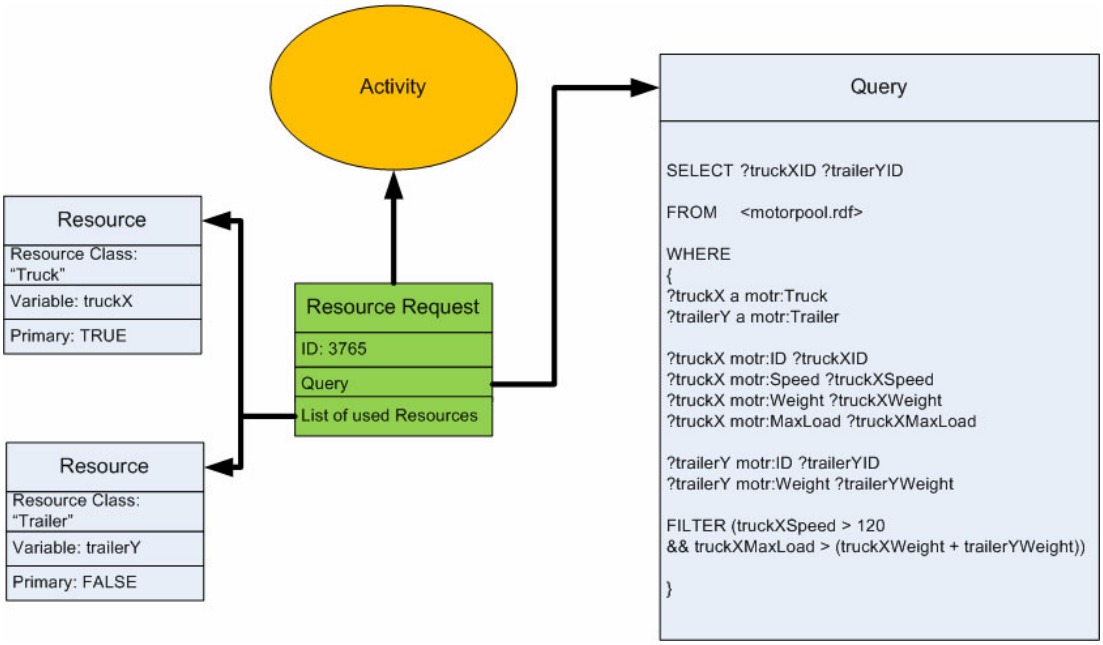


Figure 19: Example Resource Request

4.2.3 Scopes

Sometimes the interaction between a group of activities is very strong and they have to be executed in a interdependent sequence. It is, for example, possible that a couple of activities need the same resource object and if the resource object is reassigned anywhere else during the execution of any of the activities the whole operation has to be begun again from the beginning.

For this purpose it is necessary to define a group of activities. Following BPEL this grouping is called a scope [Pelt03]. When the process enters a scope, i.e., when for one of the activities that is inside the scope, the execution has begun, then a certain action can be performed. This action can, e.g., be the blocking of the resources for any activity outside of the scope. Similarly when the last activity of the scope has been executed, an action can be performed.

Scopes are very suitable to make sure resources are reserved or allocated as well. Both on entering the scope and on leaving the scope resources can be reserved for other resources that are farther in the future. This is an especially vital benefit in cases where only one of several paths is actually executed. In this way resources can be reserved or allocated even though it is not know beforehand which path the process will take.

4.2.4 Process/Resource Rules

To govern the complex relationships that exist between the resources and the process model, the modeler needs a wide variety of tools at his disposal. This is achieved through the different process/resource rules that govern the interaction between the process and the resources. Most of these rules are either global or local rules. Global rules are defined for a resource and apply regardless which activity is affected. Local rules, in turn, can affect a single activity, all the activities of a scope or all the activities of a process. Thereby, the more specific rules override the less specific rules. Therefore, if a process-wide rules exists, the respective global rule is ignored and if a rule for a single activity exists the scope-wide rule is ignored.

Both the local and the global rules are hard rules, meaning they can't be ignored or modified by users of the system without changing the process model.

1. Resource Availability Rules: The availability of a resource object can be controlled in a number of different ways. The global availability rule defines how the availability state changes depending on a number of different *triggers*. Availability rules are defines as a series of predicate logic statements that can link any action of the process management system to the availability state of a specific resource. A availability state change can in this way be triggered by the execution of an activity, by the change of the property of the resource object or by the reservation of a completely different resource object.

   Local availability rules, in turn, can define different sequence of availability states and make sure that the resource is handled in the correct way if the situation demands e.g. longer waiting times before the reuse of a resource. The transition between the different availability states is one of the most important basis for the interaction between the user and the system. The most basic use of the resource availability rule is the change of the availability state when an activity is executed. The availability state of all resource objects that are used by the activity is automatically set to occupied at the beginning of the execution. Similarly a resource objects is automatically released and the availability state set to available when the execution of the activity currently using the resource object is finished. Another very important One way that both global and local resource availability rules are used is in the automatic reservation and allocation of resources. Thereby, the execution of one ore more activities is used as a trigger to reserve or allocate a specific resource object.

   The availability rule itself can include references to the resource properties. In this way it is for example possible to realize multitasking resources that only become unavailable if they are used by a certain number of activities at the same time. Additionally the availability rule can refer to the availability state or attribute of another resource. The resource availability rule makes sure that the modeler is as free as possible to model the real transformations that the resource undergoes.

2. Process Prioritization Rule: In a world of limited resources it is important to ensure these resources go to the activity that needs them the most. In cases where different

processes request the same resource there has to be a way to determine which process gets the resource. Instead of relying on a simple first-come-first-serve strategy every activity, scope or process can be assigned a priority and the request with the highest priority will receive the resource. Since the priority is directly connected to a local entity (activity, scope or process) a global process prioritization rule does not exist.

The priority is a set of four numbers between 1 and 10 which signify the priority depending on which availability state the resource is in. The first number is the basic priority. This comes into play if the resource is available. The other three numbers come into play if the resource is either reserved, allocated or occupied by another activity. If an available resource object is requested by different activities at the same time, the basic priority of the activities are compared and the activity with the highest priority gets the resource object. If a resource object that is either reserved, allocated or occupied by an activity, is, e.g., reserved by another activity the basic priority of the activity that currently "owns" the resource object is compared to the priority (reserved), priority (allocated), priority (occupied). Again the highest priority gets the resource object. Normally, the basic priority of an activity is highest, priority (reserved) second highest, priority (allocated) third highest, and priority (occupied) lowest. In this way the more firmly an resource object is "spoken for", the lower is the probability that it is divided from its current owner.

Giving activities the power to interrupt execution of other activities has of course major potential for chaos and an environment where many process steps have to be halted because a resource object has been requested by a higher priority activity. Giving a resource a high priority (occupied) should therefore be restricted to activities that are so time-critical that they cannot be delayed under any circumstances. It is common not to assign a separate priority for every activity of every process. In this case the priority of the process is used. If no priority is assigned to a process, a standard priority is used which is the same across the entire system.

Example 5 shows how a resource object is affected by different activities with different priorities.

Example 5 (Process Priority Rule)

In this example three different activities with different priorities seek to use a single resource object.

In the beginning the resource object is available. It is then first reserved by the low-priority Activity A. Since no other activity currently requests this resource the resource is reserved in **step 1**.

After a short time Activity B also requests the resource. The basic priority of Activity A is compared to the priority (reserved) for Activity B. Since Activity B's priority is higher the

resource is transferred to the reserved state, but this time for Activity B in **step 2**. Since there are no further competitors the resource is allocated to Activity B in **step 3**. Finally the resource object is occupied as Activity B begins its execution in **step 4**.

During the execution Activity C requests the resource. The basic priority of Activity B is compared to the priority (occupied) of Activity C. Since Activity B's priority is again higher, it can complete its execution and afterwards the resource is transferred to available again in **step 5**.

Now both Activity A and Activity C request the resource to be reserved and their basic priorities are compared. Since the priority of Activity C is higher, the resource is reserved for Activity C in **step 6**.

A graphical representation of the sequence of events is shown in Figure 20. The upper part shows the availability state of the resource object, the middle the sequence of requests from the activities, while the bottom part shows the different activities with their priorities.

3. Resource Modification Rule: Often the use of a resource by an activity is accompanied by a change in one of the resources properties. Sometimes this is a change of quantity in a raw material, other times an additional capability of a human resource at the end of a management training course. The execution of processes often has a direct effect on the resources. The resource modification rule governs this effect and how the properties changes as a result of the execution.

   Similar to the resource availability rules the resource modification rules are defined by a series of predicate logic statement. The changes in the properties of the resources are triggered by an event of the process management system that can (but does not have to) be related to another resource object. The resource modification rules in this way work as a sort generalized resource availability rules that can effect changes not on the availability state, but on any user-defined property of the resource object.
   Resource modification rules also exist as both global and local rules.

4. Resource Substitution Rule: The ideal resource for a certain task is not always available. Employees take vacations, machinery is lent to another department and various other unforeseen changes affect the resources of an organization. Resources that are in high demand are often fully booked for weeks or months. Many of these problems cannot be anticipated beforehand and therefore cannot be included directly in the resource request. Sometimes the knowledge of the person who models the resource request does not extend to the way work is distributed inside the departments and working groups.

   For these cases substitution rules make it possible to fulfill request that would normally be unattainable. A global substitution rule for a resource specifies a resource that can be used if the original resource is unavailable. A request for this resource simply returns the substituted resource instead of the originally requested one.

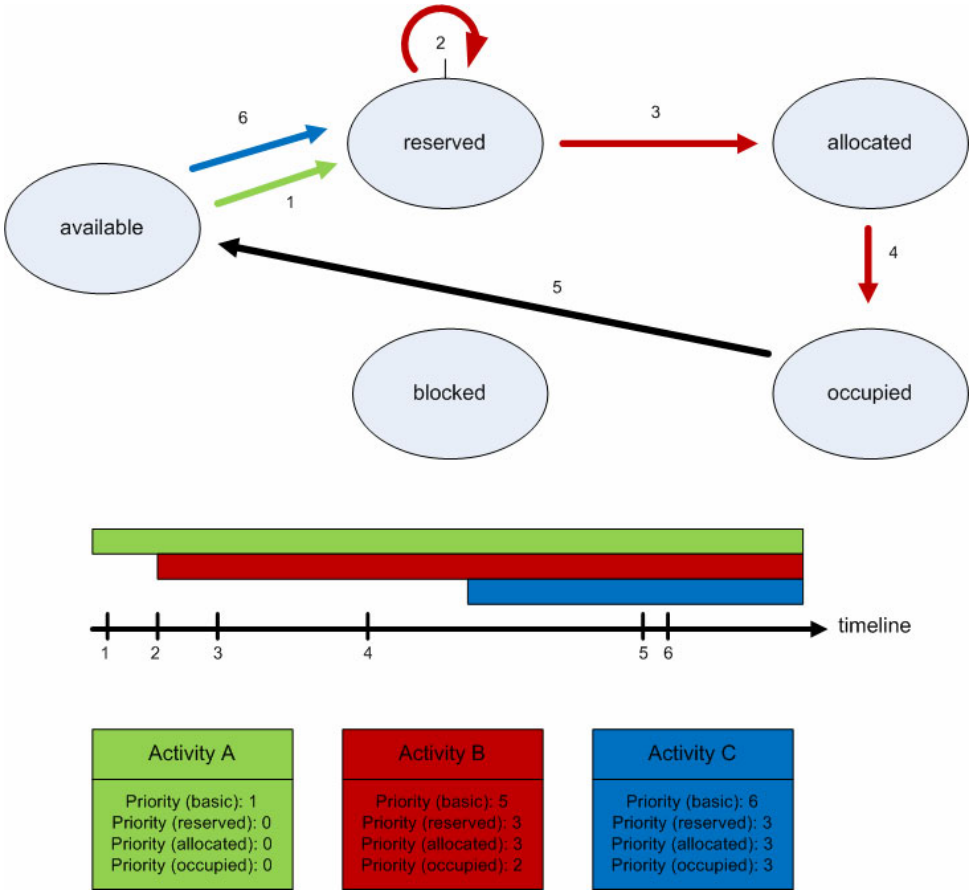


Figure 20: Process Prioritization

In addition to a global substitution it is also possible to define local substitutes that only are used when the request is from a specific activity, scope of activity or process task. This makes sense in cases where the modeler has additional knowledge about the activity and the requested resource and knows that certain properties of a resource will not be needed and therefore a wider range of possible substitutes is available.

Both with local resources and global resources it is important to make sure that the substitute can do the same things that the original resource is able to do. Especially as a global substitute it is unclear which activity will request the resource and if the properties of the substitute deviate too much from the original resource problems can arise. If a local substitute and a global substitute exist for a resource then in case the resource is unavailable first the local substitute is requested. If that is unavailable as well then the global substitute is used.

5. Resource Access Rules: Not every department has access to every resource in the company and some resources have to be restricted altogether. Resource access rules make sure that only the processes that have the proper authorization can access certain resources. For every resource it can be defined by which users or user groups the resource can be allocated and used. Primary resources are obviously unrestricted by these access rules, since they are allocated by the system. If a resource request contains resource objects to which the primary resource doesn't have access the system outputs an error message and gives the primary resource the opportunity to let the additional resource be requested by someone with the necessary access privileges.

6. Resource Selection Rule: In contrast to the process prioritization rule where multiple processes compete for a single resource, it can occur that an activity has more than one resource to choose from as a result from a resource request. While most query languages offer ways to sort the results of a request these are not mandatory to use and don't always provide a single "first place". In these cases selections rules define which resources should be chosen. Commonly only one resource selection rule for the complete system is defined, but it is also possible to define separate local rules for scopes or even activities. The rules simply define by which properties the resource objects are sorted in the case of multiple matches.

4.2.5 Resource-based Process Constraints

Many processes make great demands on the interplay and relations between different resources. Legal requirements like the Sarbanes-Oxley Act [SarOxA] and internal guidelines require two-person-integrity for special tasks. Additionally, different tasks of a process sometimes should be performed by different persons to make sure errors get spotted and no manipulation can occur. In other cases, it may, for example, be important that the customer does not have to talk to a variety of different people and is in contact with the same customer agent on every process step. Even with non-human resources these types of requirements can occur. For example, the end-product of one machine can only be processed by a certain machine while the end-product of another machine may be used by all machine types.

The way to integrate these requirements into the process/resource model is by using additional constraints. Constraints link two or more different process steps and add a certain requirement to the resources that are involved. The requirement is a Boolean expression that governs the identity or disparity of the resources or defines a more complicated relationship between them. Nodes and individual resource requests can be filled with more than one constraint and in this way any number of resources can be linked.

There are two different types of constraints, hard constraints and soft constraints. Hard constraints are requirements that have to be met regardless of individual circumstances and cannot be ignored without a severe penalty. This often is the case for legal requirements or internal standards that are deemed of utmost importance. If the constraint cannot be met, the process step cannot be executed until a resource that fulfils the constraint is available. Hard constraints are therefore a very powerful tool and have to be handled with care by the

modeler. Incorrect or excessive use of hard constraints can lead to long waiting times and deadlocks.

Soft constraints on the other hand can be used for requirements that should be followed but that can be by-passed if the situation requires it. The user will be informed that a certain constraint could not be fulfilled and will have to decide whether to wait for the optimal resource or continue with a resource that does not adhere to the constraint. Depending on the privileges of the user this decision may have to be made by a superior. Soft constraints often represent best practices or solutions that are superior to others but not strictly necessary for a satisfactory execution of the process. They constitute an easy way to identify the ideal resource composition for a process and leave it to the user to decide if a certain goal is worth waiting for a heavily requested resource.

The restrictions expressed through constraints are integrated into the resource request by the system. The resource objects that are touched by the constraints are often unknown at the time the constraint is designed. Additionally, constraints usually refer not only to one single activity, but to multiple activity. Therefore, placeholder objects are inserted into the query which signify the resource objects of another activity. Once these resource objects are resolved, they can be clearly identified in the requests of other activities and therefore used to make the selection.

5 Resource Execution

After the resources and the integration between resources and process steps are designed the next phase is the *execution phase*.

At the time of the process execution, it is the task of the system to carry out the decisions that have been made at modeling time. The resource requests have to be matched to the real resources and the allocation and return of the resources has to be managed.

In chapter 5.1 the requirements for the resource execution are presented. Chapter 5.2 outlines the different concepts in play during the execution phase. Chapter 5.2.1 explains the reservation and allocation, 5.2.2 the matching and 5.2.3 the claiming, use, and release of resource objects. Finally, Chapter 5.2.4 discusses the role of resources in adaptive process management.

5.1 Requirements Resource Execution

While the requirements for the execution phase of course are related to the requirements for the previous phase, nevertheless there exist some issues that are mainly relevant in this stage.

1. Timely Reservation and Allocation of Resources: In a system where a great number of different processes interact with many resources it is important to know if a specific resource is planned to be used by other resources in the near future. For this purpose resources should be reserved in advance to make the behavior of the system more predictable.

2. Fast and Efficient Matching of Resource Requests and Resource Objects: The matching of resources and resource requests is at the heart of the process execution from a resource perspective. The matching system has to be able to work with many thousands to millions of resource objects and a lot of simultaneously running processes. It has to be fast enough not to pose a significant delay in the work process of the users of the business process management system.

   The matching protocol obviously is highly dependent on the used data model and query language. The software component for executing the matches should be as independent from the business process management system as possible. Ideally a separate matching server can be used.

3. Support for Adaptive Process Management: Unexpected deviations from the planned sequence of events are common occurrences in any organization with complex business processes. As business process management systems are developing solutions for these cases, it is important that the integration of resource management can carry out the dynamic changes that are necessary.

5.2 Resource Execution Concepts

The way of dealing with resources in the execution phase flows directly from the decisions made in the modeling phase. Reservation, allocation and matching are based on the resource requests. Additionally some issues occur in the execution phase that do not have a root in the modeling phase.

5.2.1 Reservation and Allocation

Both reservation and allocation are handled by a manipulation of the availability state of one or more resources. The difference between reservation and allocation is one of intent and of time. Reserving a resource means that a process plans to use it in the future and wants to make sure that other processes know this and are able to plan accordingly. Reserving a resource can be tentative, based on statistics about past process executions. Allocating a resource means that the use of the resource is firmly scheduled and imminent. Taking over an allocated resource-which is only possible if the allocated priority of the taking resource is higher than the basic priority of the original resource-seriously disrupts the process execution and should be a well-justified exception and not standard operating procedure. There are two different ways that reservation and allocation of resources takes place in the system.

The first one is a manual reservation/allocation by an user with the authorization to change the availability state. Usually this user is the primary resource for an activity. The user decides when he has enough information about the presumed process flow that it makes sense to reserve a specific resource object. He is generally free to change the availability state of resource objects. Resource requests are able to abstract from the resource objects and describe only needed features of a resource without committing to an unique resource. In contrast reservation and allocation of resources only work if the user knows which individual resource is needed.

Since it is often not possible for the user to know which resource will be needed, he is faced with a dilemma. If he decides to forgo reservation the needed resource might be in use by another process and he will need to wait for the resource to be "unfrozen". On the other hand if he prematurely reserves a resource it might not be needed after all because either the process takes a different route than expected or the matching process returns a different resource than the user expected. This results in delays as other processes can't use the reserved or allocated resource.

There is no easy solution for this dilemma. It depends heavily on the number of active processes, the organizational structure and the number of critical resource objects among

other things. There is also significant intersection with the process prioritization rules in play. The more the basic priorities differ from the priorities for reserved resources the bigger the effect of a reservation is. The optimal way to handle this can only be found on a case by case basis and will require experimentation.

Because of these difficulties, the more comfortable way to handle reservation and allocation is using the resource availability rules for the resources to change their availability state based on events during process execution. This allows for a fine-grained reservation policy based on the execution of certain process steps, the selection of certain resources or the start of a process. For every resource object multiple predicate logic statements are assembled that exactly define under what circumstances the availability state of the resource object changes and if it is reserved or allocated. The differentiation between local and global resource availability rules allow the reservation or allocation to take place in different ways depending on which activities are involved.

In the case of raw materials the reservation and allocation is handled in a slightly different way. Since raw materials don't stand for discrete items, but a certain amount of something, it makes no sense to block the complete resource object, because it is reserved by another activity. Therefore, if a certain volume of the material is reserved, the "quantity" property of the raw material object is decreased by that amount and the "reserved" property of the object is increased by that amount. If the reservation is cancelled then the "reserved" property is decreased and the "quantity" property is increased again. Similarly, if a volume of the material is allocated, the "allocated" property of the object is increased, while the quantity is decreased. In this way the raw material stays available for use by other activities until the amount of usable raw material reaches zero.

In practice it can prove to be beneficial to use a combination of manual reservation and reservation by resource availability rules. A reservation policy usually does not cover every resource but mainly resources and processes that are in high demand by different processes and behave in a predictable manner. The rest of the resources are reserved manually or used without any reservation at all.

Often a change of the availability state from available to reserved or from reserved to allocated results in changes to the physical resource. Sometimes the preparation of a machine can take a long time and it would waste valuable time to start preparing it only if the availability state changes to occupied. Therefore depending on the organizational structure the preparation can start when an object is allocated or even reserved. Sometimes a resource object is not at the location where it is needed for execution. In these cases the object can be moved when it is put in the reserved or allocated state.

5.2.2 Matching

The matching protocol is the heart of the resource management in the execution phase. It takes the resource requests linked to the process steps and compares them to the available resources in the resource pool. While taking into consideration the rules governing resource

selection and access the request is parsed and a list of suitable resource combinations is displayed for the user.

There are a number of existing matching servers for main data models like RDF/SPARQL that fulfill the requirement of a fast and efficient matching of resource requests to resource objects. In our case this is be achieved by the Apache Jena Framework which includes a SPARQL query engine [ApaJen]. The query engine can take over the standard matching of resource requests to resource objects. The constraints and the various rules discussed in chapter 4.2.4 on the other hand have to be integrated separately.

When a resource request is sent to the matching server, first the placeholders used by constraints and resource rules are resolved and replaced with the actual resource objects they refer to. Then the SPARQL query is matched against the RDF graph that represents all resource objects in the resource pool. The query engine then puts out all the combinations of resource objects that fulfill the query.

The matching server does not exclude results based on the availability state of the resource object. Therefore, the system filters out all combinations containing resource objects that are blocked. Then combinations containing resource objects that are in the states reserved, allocated and occupied are examined based on the priorities of the requesting activity and the activity that currently "owns" the resource. If the priority of the requesting activity is lower, the combinations containing the object will be filtered out as well. The resource combinations then are sorted according to their resource availability states, with available resources first, then reserved, then allocated and finally occupied.

Before a resource object is filtered out by the above mechanism, there will be a test, if there exists a substitution for the object. If there exists either a global substitute or a local substitute defined by the resource substitution rule, the substitute's availability state and priority will be tested in the same way described above. Only if the substitute is unavailable as well will the resource combination be filtered out, otherwise the substitute simply takes the place of the requested resource object.

For every resource object that is returned by the query mechanism, the systems checks if the requesting user has the authorization to demand it. Combinations containing a resource object for which the user does not have authorization are filtered out, after examining if the user has authorization for the substitute of the resource object. Alternatively the user can, of course, contact a superior with sufficient privileges to approve the interaction.

Finally, if more than one combination of resource objects is returned to the user, he has to decide which resources fit his purposes the best. The combinations are presorted using the resource selection rules and the soft constraints, but the user is not obligated to pick the highest-ranked combination. If the process step is an automatic one that does not involve an user to make the decision, the system simply takes the highest ranked resource combination.

Example 6 shows the matching process for a typical resource request.

Example 6 (Matching process)

This example shows the matching process for a resource request that includes a constraint and the use of resource substitution rule.

At the beginning of the process stands the activity with accompanying resource request, that we have seen in Figure 19. The resource request specifies a combination of two resource classes, one "Truck" with a speed of 130km/h and one "Trailer". The maximum load of the truck must be bigger than the combined weight of truck and trailer. After the request is defined, an additional constraint is put on the truck. The resource objects which fulfills the request cannot be the same truck as the one that fulfills an (identically framed) request two steps before the current activity.

At the execution time of the activity, the reference to the other truck is resolved and identified as "Truck No. 78263". After the placeholder is resolved the resource request is send to the matching engine. The matching engine compares the resource object in the resource pool to the request and comes up with three different resource combinations:

1. "Truck No. 38412" and "Trailer No. 1934"
2. "Truck No. 98243" and "Trailer No. 3214"
3. "Truck No. 98243" and "Trailer No. 1934"

As seen here, it is possible that the same resource object can be part of different resource combinations. The next step is to check the availability state of all resource objects returned by the matching engine. "Truck No. 38412" is currently in use by another activity and the priority of that activity is higher than the activity that sent the request. Since "Truck No. 38412" does not have a global or local substitute, the first resource combination is filtered out. "Truck No. 98243" is available, but "Trailer No. 3214" is currently reserved by another activity with a higher priority. In contrast to the truck, "Trailer No. 3214" has a substitute. This substitute is "Trailer No. 1934". The resulting combination is, of course, the same one as the third combination in our list and is therefore filtered out as a duplicate.

Consequently, the third combination, "Truck No. 98243" and "Trailer No. 1934" is the only one that satisfies the request. The execution of the activity is therefore started with "Truck No. 98243" as primary resource and "Trailer No. 1934" as additional resource.

5.2.3 Claiming, Use and Release

After the user decides which combination of resources he wants to use, he has to claim the resources. This takes place by again checking if the user has access privileges for every resource object in the combination. If he has, then the availability states of the resource objects will be set to occupied through the use of the resource availability rule and the work on the execution of the process step can start.

There are a number of things that can prevent the proper execution of a process step. An activity can be stopped because it is no longer needed or declared a failure by either the user himself or a superior. A resource can break or malfunction. A resource object can be demanded by an activity with higher priority and the complete execution has to be stopped. In all these cases the user has to make a decision how to proceed. There are a number of different ways that he can continue.

In cases where he sees that he is unable to execute the process step in a reasonable timeframe, he can simply declare the activity as failed and release the resource objects. Depending on the modeled process this can result in the failure of the complete process or an alternate route that is taken. The availability state of the resource objects is automatically put to available by using the resource availability rules and they can be reserved and used again.

If a single resource object has to be blocked because of malfunction or other reasons during the execution of the activity, the user can simply send a new resource request for the missing resource. This works in the same way as a regular resource request, but only for the now unavailable resource. If the request results in a new resource, then the planned execution of the activity is resumed. If no alternate resource can be found the user has to decide, if he waits for the original resource object to become available again or if he declares the activity as failed. If he waits, all other resource object stay in the occupied availability state and are not available for other activities. Otherwise all resource object are released similarly to the previous paragraph.

If a resource object is demanded by an activity with a higher priority, the object is transferred to that activity and the currently running execution is put on hold. The other resource objects stay occupied. The user can then send a new resource request for the missing object and continues in the same way as he would if the resource had been blocked.

If the activity completes the execution normally then the resource objects are freed and their availability state put back to available through the use of the resource availability rule. They then can again be requested normally.

5.2.4 Resources in Adaptive Process Management

Making it possible to change process models during the time of execution both at the level of process instances and with regard to the process types is a prerequisite for a business process management system that aims to fully capture the complexities of an organization.

As Reichert [Reic00] has shown, a significant number of processes are so unpredictable that it is impossible to gather all the data needed to model the process correctly before execution time. A business process management system therefore has to include the capability to handle these unpredictable processes if it doesn't want to omit a significant part of day-to-day business. Consequently the resource management component of a business process management system has to support these ad-hoc change as well.

5.2 Resource Execution Concepts

In the event of ad-hoc changes of individual instances, the availability of resources is very important. If, for example, an additional process step is added to the process instance it might be the case that this process step needs resources to execute. After the process step is added according to the rules of the business process management system (see, e.g., [DAG+06]) the user can design and send a resource request in the same way as in the process modeling phase.

Ad-hoc changes often occur because of unforeseen events that have to be dealt with immediately and there is normally no time for reserving or allocating the resources. The newly added activities should therefore be assigned a higher priority than normal, to ensure that the resource request can be fulfilled.

When a activity is deleted during execution, the resource request is cancelled and any reservations and allocations that have been made for this activity are reversed. Similarly if an activity is moved to another place in the process, the reservation and allocation are cancelled. In this case however the resource requests are retained, since the activity still exists.

The ad-hoc change can also be restricted to the resource component itself. The requirements for the activity might have changed or the user decides that the resources specified by the modeler are not sufficient for the execution. If he is authorized to make ad-hoc changes to the process, he can invalidate the original resource request and change it or design a new request in the same way as it is done in the modeling phase. Rules and constraints can similarly be changed during execution.

The second important component of adaptive process management is the support of process type changes and the propagation of these changes [RRKD05]. If a process has to be changed because the structure of the organization has changed or a new approach is used to solve a problem, this involves all the instances of this process that are being executed. While it may be possible to complete some instances on the basis of the old process, oftentimes this will not work. Therefore, these instances have to be "migrated" to the new process without disrupting the running execution [DAG+06].

Obviously, the migration of running instances to a new process can result in significant changes to the resources that are involved. The resource requests themselves remain relatively unaffected, because they simply are moved, deleted or inserted with the activities to which they belong. The reservations and allocations on the other hand have to be fixed either by hand or by using an algorithm that determines up to which activity the execution remains unchanged and then keeps the reservation that are not touched by the migration. The reservations that fall in areas, where change of the process occurs, have to be examined and cancelled if they don't fit to the new process anymore. Similarly, resource constraints and resource/process rules have to be adjusted to fit the migrated process instances.

6 Resource Monitoring

In the context of business processes, the more knowledge about the execution and structure of the processes is available, the easier it is to optimize the processes and the closer the processes can approximate the real work routine. For that reason, the monitoring and logging of processes and the analysis of the resulting information has become a significant component of business process management systems. Additionally the field of business intelligence has come into being to take advantage of the data being produced by business process management systems and other software that logs its progress. Information about resources and their use, reservation, and allocation is also very valuable in this context. It can help to optimize the purchase of new machinery, keep idling and downtime to a minimum and help employees to work together in a more effective manner.

In order to collect the right kind of information and make good use of it, the system has to fulfill a couple of requirements. These are developed in chapter 6.1. In chapter 6.2 the individual concepts are listed, with chapter 6.2.1 examining the resource logging itself and chapter 6.2.2 showing in what way business intelligence can benefit from information about resources. Chapter 6.2.3 shows how the gathered data can predict the usage of resources and supply operational decision support.

6.1 Requirements Resource Monitoring

The requirements for resource monitoring result directly from the "comprehensible monitoring and logging" requirement put forth in chapter 2.3.

1. Logging of resource usage: All information that is available about the interaction of the business process management system with resource objects should be logged and saved. This includes the complete logging of the resource calendars which includes information about all property changes that affect the resource and the activity or user by which the change has been initiated. In this way the reservation, allocation and use of the individual resource objects can be traced.

2. Provision of Real-Time Data: Data about the current availability state of the resource objects should be made available in real time. It should be possible to see the current resource requests and understand where the resource conflicts lie. Similarly the state of every process should be documented so a user can see, e.g., all processes that are currently waiting for a resource. This data can be very helpful for users to decide when

problems arise and can give system administrators information to help resource conflicts.

3. Utilizing Past Usage Patterns for Predictions about Future Process Executions: As shown in chapter 5.2.1 the manual reservation of resource objects is prone to error and difficult to manage. To utilize an automatic reservation by using the resource availability rules, it is important to have reliable information about past usage of resources. This information can then be used to make predictions about the probability that a resource object will be used in a future execution and thereby decide if it makes sense to reserve or allocate it automatically.

6.2 Resource Monitoring Concepts

The amount to which these features can be implemented is again highly dependent on the business process management system. Especially the more complex components don't just concern the resource part of the system, but act in concert with the business intelligence capabilities of the business process management system.

6.2.1 Resource Logging

The main way in which information about the resource objects is saved are the *resource calendars*. In these every change to the objects is documented in a way that allows complete traceability of the actions working on the objects. The individual changes to the resource objects that are documented in the resource calendars are called *events*.

For every event a number of data points are saved. These include a timestamp, the property which is changed, the value that the property takes on and the activity or user that caused the change. The changes of all different resource objects in the resource pool are saved together in a common log file, the *resource calendar log*. The individual resource objects can be identified by their name.

Example 6 shows the way in which these events lead to a comprehensive resource calendar log.

Example 6 (Resource Calendar Log)

In this example the relationship between the events concerning a number of different resources and the resource calendar log are presented.

A driver, identified as Driver No. 4693, discovers that the maximum load for the truck with number 74312 is incorrect. Instead of the given maximum load of 11.600 kg the real maximum load of the truck is only 6.600 kg. The driver corrects this by changing the attribute value and this results in the first entry in the log file. He then discovers that the trailer that is currently attached to Truck No. 74312 is too heavy for it and attaches the trailer to another

truck, this one with the number 23654. This change results in two inputs into the log file, the change of the property "Attached Trailer" to NULL in the Truck No. 74312 and the change of the property "Attached Trailer" to "Trailer No. 6574" in the Truck No. 23654.

Because all changes to the properties have been done manually, the driver with the number 4693 is given as the cause of the entree. After some time, both the truck and the trailer are reserved automatically when a process instance reaches the activity "SelectTruck" due to the resource availability rule of the truck. The availability state of the Truck No. 23654 and of the Trailer No. 6574 are changed to "Reserved". In this case the reason for the change in the log is not an user, but the activity which caused the change: "SelectTruck".

This sequence of events results in the resource calendar log shown in Table 2.

Resource Object Name	Changed Property	New Value	Timestamp	Changed By User	Changed By Activity
Truck No. 74312	Max. Load	6.600	2008-11-29 10:45 UTC	Driver No. 4693	
Truck No. 74312	Attached Trailer	NULL	2008-11-29 11:00 UTC	Driver No. 4693	
Truck No. 23654	Attached Trailer	Trailer No. 6574	2008-11-29 11:05 UTC	Driver No.4693	
Truck No. 23654	Availability State	Reserved	2008-11-30 23:01 UTC		SelectTruck
Trailer No. 6574	Availability State	Reserved	2008-11-30 23:01 UTC		SelectTruck

Table 2: Resource Calendar Log

All resource requests that are executed are also logged in the resource request log. The log includes the unique ID for the request, the variables with their respective resource class and the query. For every entry a timestamp is added as well as the state of the request. The state of request can be either "designed", "in execution", "completed", or "failed". Example 7 shows, how different requests are saved in the resource request log.

Example 7 (Resource Request Log)

In this example it is shown how resource requests are stored in the resource request log.

The log begins when a modeler designs resource requests for two activities, activity A and activity B. The request for activity A contains only one resource class, a "Truck" with the

variable "truck". The ID of this request is 743496. The resource request for activity B is composed of a "Truck" with the variable "Truck C" and a "Trailer" with the variable "Trailer D". Its ID is 925643. For both resource requests an entry is made in the resource request log, with their respective unique IDs, the variables used, and the query. For both resources the state is "designed".

After a while, the modeler decides to change the query of activity B. This results in a new entry for the request with ID 925643, identical to the first one, but for the timestamp and the query text.

Then activity A begins execution and another entry is added for request 743496. This one differs from the first one by the timestamp and the state which is now "in execution". One final entry is then added to the log when activity A is completed successfully, this time with the state "completed". This sequence of events results in the resource request log shown in Table 3. To save space only the FILTER clause of the query is shown here.

Unique ID	Activity	Variables	Query	Timestamp	State
743496	"A"	{truck: "Truck"}	FILTER truckXSpeed >120	2008-10-23 15:45 UTC	"designed"
925643	"B"	{Truck C: "Truck", Trailer D: "Trailer}	FILTER truckCSpeed >120 trailerDWeight < 5000	2008-10-23 15:53 UTC	"designed"
925643	"B"	{Truck C: "Truck", Trailer D: "Trailer}	FILTER truckCSpeed >110 trailerDWeight < 5000	2008-10-23 16:21 UTC	"designed"
743496	"A"	{truck: "Truck"}	FILTER truckXSpeed >120	2008-10-23 17:08 UTC	"in execution"
743496	"A"	{truck: "Truck"}	FILTER truckXSpeed >120	2008-10-23 18:21 UTC	"in execution"

Table 3, Resource Request Log

6.2.2 Post-Execution and Real-Time Analysis

The data which is gathered by the resource calendar log and the resource request log can be used to provide accurate data, not only for post-execution analysis, but also for immediate information of users and system administrators. There are a number of possible applications for different parts of the log.

1. Listing of all resources and their current availability state: By taking only the last entry in the log for every resource object one gets a detailed breakdown of the workload of all resources used in the organization. It is easy to assemble this data into statistics which provides the percentages of available, occupied, reserved, allocated and blocked resource objects.

   It is possible to define critical situations out of these statistics, so a manager is e.g. informed if the percentage of non-available resources in the whole organization - or a specific section - goes above a certain threshold. In this way resource conflicts that could not be identified at the individual level can be seen by looking at the complete picture.

2. Identification of unnecessary ownership changes of resource objects during process execution: The process prioritization rules allow activities with high priorities to claim resource objects which are already reserved or even in use by other activities. While this can be necessary to guarantee a timely execution of especially important process steps, it can also lead to unsatisfactory situations where an activity repeatedly has to stop execution because a single resource object has been claimed by another.

   By looking at sudden and repeated changes to the availability state for resource objects these situations can be identified. In this way the activities which are causing these changes can be identified as well as the activities which are repeatedly victimized by them. Often a small change in the priorities of either the causing or victimized activities can ensure that the problem will not occur again.

3. Detection of incorrect property changes: The resource modification rules gives the activities a large degree of freedom if they want to change properties of resource objects. This allows the automation of complex procedures with regard to several resource objects. Unfortunately this also can lead to errors, which are difficult to find, if the rules aren't handled correctly

   Since several activities as well as individual users and modelers can change the properties of resource objects the modifications are often hard to comprehend on a micro level. By using the resource calendar log and displaying the changes to a single resource object or even a single property, problems can be easily identified.

In addition to the resource calendars one can also take the information from the resource request log. With this and in combination with the information from the resource calendar log other problems can be solved.

4. Analysis of requested resources: The resource request log makes it possible to see which types of resources are requested how often, which resource objects are requested directly and which resources are often requested together. This information can be used to identify shortages of special types of resources and can suggest which resources can be aggregated to combined resources.

   It also gives a good overview how the resources are requested in the organization. Requests which mostly ask for specific resource objects for example can mean an organizational structure with many heterogenic resources that can't be substituted for each other. If the actual organization doesn't fit this description though, it is important to find out, why the modelers still use these direct requests instead of more flexible abstract ones.

   Furthermore, it can be used to identify requests which are difficult or impossible to fulfill. Sometimes modelers or users don't have the necessary information about the resource pool and design resource requests that demand resource objects that are not available, don't exist anymore or can't be combined. Using the resource request log these requests can be easily identified and modified to construct resource requests that can be fulfilled.

5. Using requested resources to watch for shortages: The resource request log in combination with the resource calendar log can identify shortages of resources by looking at reservations, allocations and requests of similar resource objects. If all resource object of a certain class are reserved, allocated or in use and a number of future resource requests need resource objects of this class it is necessary to analyze if other resources can be used instead or if priorities of the activities have to be changed.

As seen above, the gathered information can be used in a variety of different ways. In addition to this, this information can be also used to predict future resource usage.

6.2.3 Resource usage prediction and operational decision support

Process information can be used in a great number of ways to make the execution of processes easier and help users decide which path to take [APS10]. In a similar way information about the past executions of processes which contain resources can help to make prediction about which resource types or objects will be used in a running process.

Process mining can be used both for an in-depth analysis of already executed processes and for real-time decision support. This can take place either as an extension to business process management systems or as an independent framework like ProM [ProM].

Using process mining it is possible to discover process models from event logs [Aals04]. If you combine this approach with information that is available inside and outside the business process management systems about the usage of resource objects it can be possible to include simple resource requests automatically in the discovered process.

6.2 Resource Monitoring Concepts

Using process mining to make predictions about future execution of processes can be very helpful to make reservation and allocation of resources easier and more reliable. Especially in an environment where identical processes are executed many times, it is possible to statistically analyze which path is taken the most and therefore which resource requests are usually executed. This information can be incorporated into the resource availability rules and used to reserve or allocate resource objects based on the usual path taken and the usual resource object used in this case.

7 Prototypical Implementation of the Resource Management Component on the Basis of ADEPT2 / AristaFlow

This section shows a prototypical implementation of the concepts put forth in the preceding chapters. The construction of the prototype is dependent on the model of the business process management system. Due to the numerous features that ADEPT2 meta model offers, especially in the field of adaptive process management, the decision to use AristaFlow as the underlying business process management system for the prototype was easy.

A complete realization of all the ideas, that have been developed in the preceding chapters, is obviously beyond the scope of this work. The prototype shows the central function of a resource management extension for AristaFlow while providing interfaces for a number of different components.

In chapter 7.1 the basic structure of ADEPT2 / AristaFlow is explained. Chapter 7.2 shows the special points of integrating the prototype into AristaFlow while chapter 7.3 presents the structure of the prototype in detail.

7.1 Basic ADEPT2 / AristaFlow Structure

The ADEPT project has been in development at the *Institut für Datenbanken und Informationssysteme* (DBIS) at University Ulm since 1995 [DaRe09]. Flowing from the OKIS project which aimed to "develop a concept for a cross-organizational, clinical information system that is able to integrate autonomous, heterogeneous departmental systems as well as to offer services across system boundaries" [DaRe09], ADEPT seeks to overcome a number of challenges which are not only critical in a clinical environment but which also are essential in a wide group of other circumstances [DaRe09].

1. Ease of use for process implementers: To make it easy for process implementers it is important that the process meta model is intuitive and expressive enough. It should be possible to abstract from the actual implementation of application functions and work with them as simple methods. To achieve this, ADEPT2 uses a graphical editor and allows "only those operations to the process implementer which allow to transform a structurally correct process schema into another" [DaRe09].

2. Ease of use for application developers: Business process management systems commonly have to interact with legacy systems that are implemented on different platforms and have varying modes of interaction. In order to make it as easy as possible for application developers, a implementation framework is needed that makes sure that the complexity of process support and ad-hoc changes do not interfere with the application development.

   This is attained by representing all application functions by activity templates that are added to the activity repository. ADEPT2 makes available different levels of abstraction which allows the application developer to design the interaction with the system in a number of different ways [DaRe09].

3. Ease of use for end users: One of the main points for achieving a easy to use interface for the end user lies in providing "sufficient level of abstraction" [DaRe09]. End users should only be called upon to make clear in what way the system is supposed to change and not have to worry about how the system achieves their demand.

   Achieving this requirement falls mostly to the application developers. It also differs much by the client used, be it the standard ADEPT2 client or a specialized client for individual situations. ADEPT2 makes the development of additional clients easy by providing an API that contains system functions for insertion of new activities or other changes to the process template [DaRe09].

4. Complex ad hoc changes and process schema evolution: The flexibility to let users deal with unforeseen events and make it possible to propagate changes made to the schema to running processes is one of the crucial parts of the ADEPT2 framework. Both ad hoc changes and process schema evolution can also be easily accomplished without deeper knowledge about the internal workings of the system. [DaRe09].

Based on these guidelines the AristaFlow project was started in 2004 to design and implement a process management system that supported both ad hoc changes and process schema evolution. In 2008 AristFlow was spun off into a commercial company that develops the AristaFlow BPM Suite.

The AristaFlow BPM Suite consists of a number of different components. The *process server* is the core of the system. It is responsible for the management of the process instances, the process templates and the *activity repository*. The basic structure of the process server can be seen in Figure 21. The *process template editor* is used to construct the process templates that will later be instantiated. The *activity repository editor* is used for the management of different activities which can be integrated and integrated into the process templates. The *OrgModel Editor* finally, manages the organization model of the system.

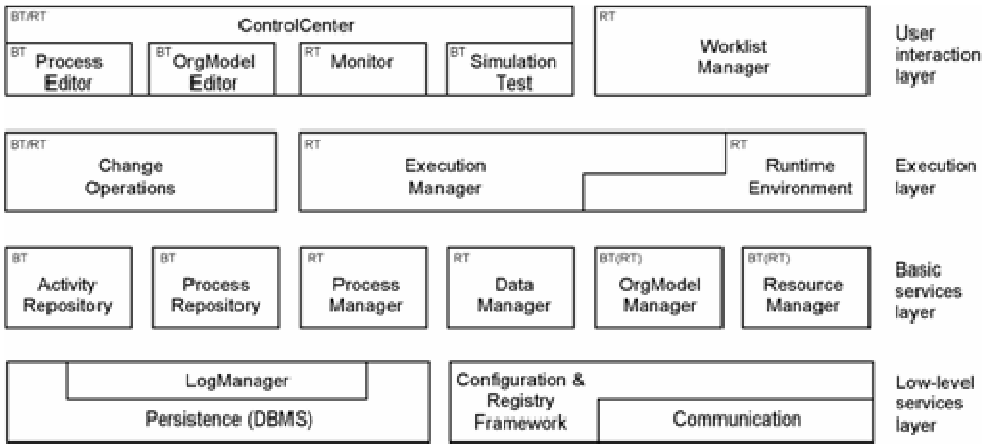


Figure 21, AristaFlow Process Server, from [AFPS]

By implementing the components in a generic way, different parts of the system can be easily accessed by other components via the API.

7.2 Integrating the Prototype into AristaFlow

As shown in chapter 4, while some parts of the resource management are relatively independent from the process management, other parts need to be very tightly integrated with the business process management system. The management of resource classes, resource objects and properties can easily be implemented separately from the AristaFlow Suite. The resource requests, constraints and process/resource rules, on the other hand, are highly dependent on it. The prototype acknowledges this distinction and is therefore divided into an independent *resource management component* and a *process interaction component* which is developed as a part of AristaFlow.

AristaFlow provides a number of features that make the development of new components easier. The detailed Javadoc, the API and the AristaFlow Plug-In Data Container make it easy to extend the functionality of AristaFlow without having to change the underlying model.

The process interaction component of the prototype works as an extension of the AristaFlow Process Template Editor seen in Figure 22 which is based on the Eclipse software development environment. The component uses both AristaFlow and Eclipse libraries including the Standard Widget Toolkit [SWT] for the interface design.

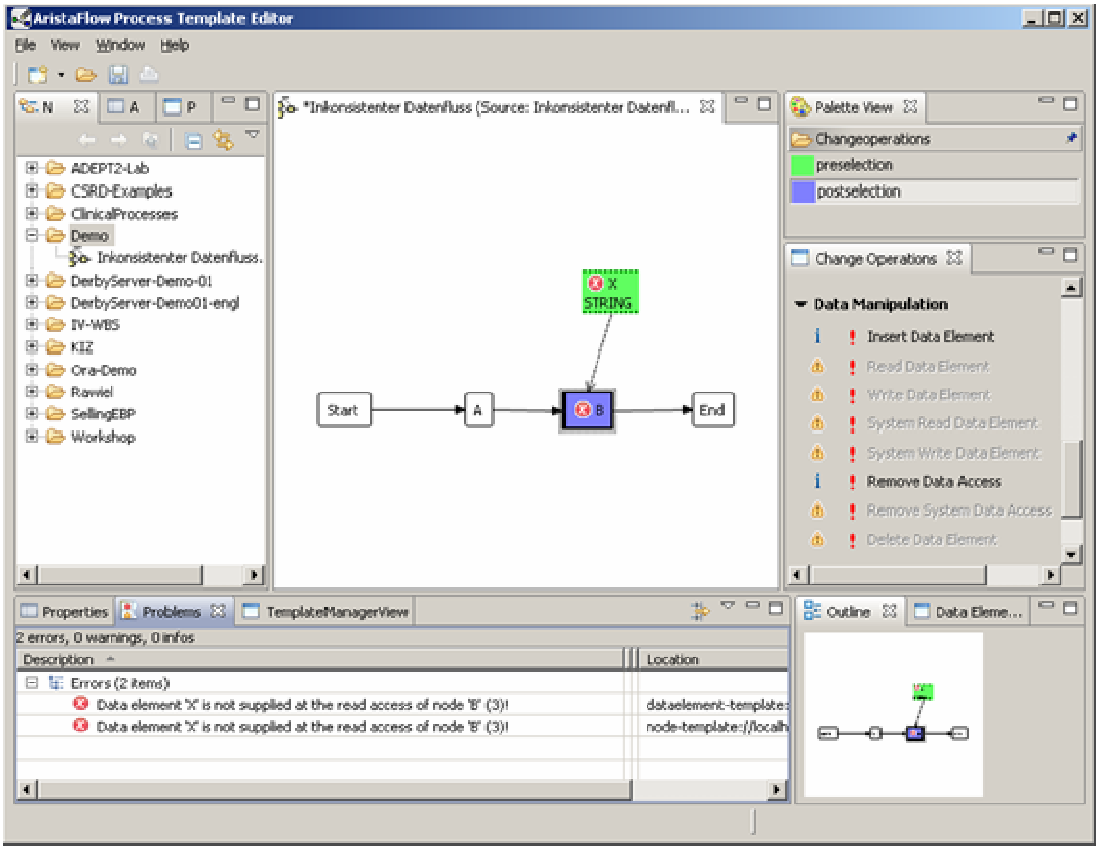


Figure 22, Process Template Editor, from [AFPTE]

The resource management component, on the other hand, uses RDF [RDF04] as a data model for the resource classes and objects and SPARQL [PrSe08] as a query language. Apache Jena [ApaJen] is used for the management of the RDF objects and as a SPARQL query engine.

7.3 Implementation Structure

As mentioned in the previous chapter the part of the prototype concerning the resource management is separated from the part concerning the interaction between resources and processes. Therefore they are discussed separately as well.

7.3.1 Resource management component

The manager, the basic structure of which can be seen in Figure 23, is divided into three different parts, *model*, *changeOperation*, and *matching.*

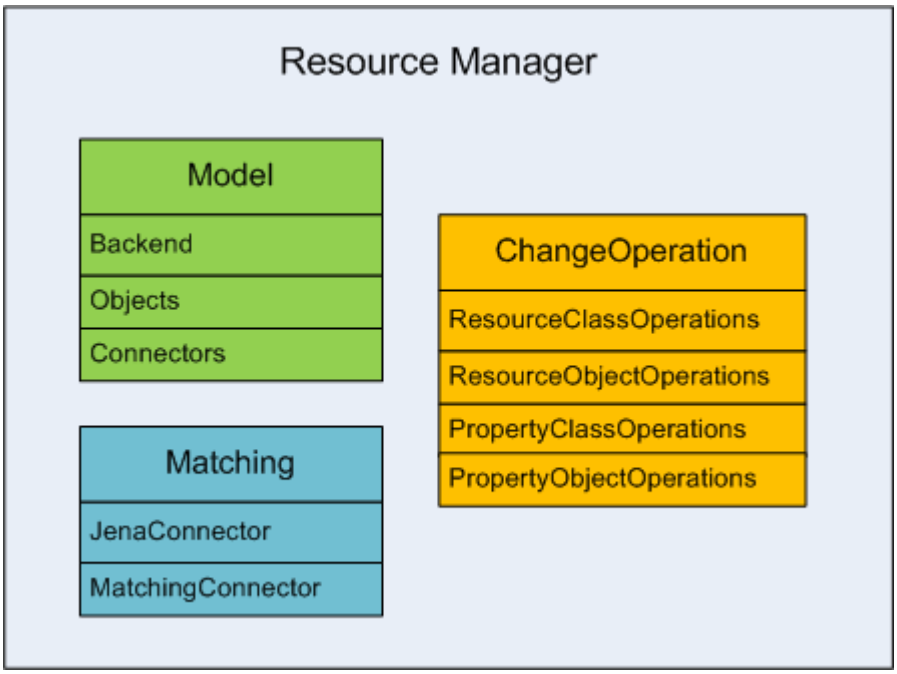


Figure 23, Resource Manager

The *model* is responsible for the underlying structure of the resources and establishes the connection to the data model, in this case RDF. It consists of the *backend*, the *objects*, and the *connectors*. The *backend* makes available the resource classes and resource objects and provides the methods for saving and loading the resource model to file. It also provides a number of different internal helper methods for other parts of the manager. Additionally it is responsible for parsing user inputs for correctness. The *objects* are the internal representations of the resource classes, resource objects and their corresponding properties. The *connectors* are responsible for the interaction of the program with the underlying data model. This is achieved by an interface that is for a start independent of the used data model. The interface can be implemented for different data models. In our case, since RDF was chosen as the data model, it is implemented by the *RDFConnector*.Every change that is made to resource objects or resource classes calls a corresponding method in the *connector* and thereby changes the RDF model. The *RDFConnector* does not directly

create or modify the RDF model but uses the Jena Framework which provides a number of different functions that simplify the interaction with the data model.

ChangeOperation makes available all operations that are needed to make changes to both resource Classes and Objects. It is again divided into four parts, ResourceClassOperations, ResourceObjectOperation, PropertyClassOperations, and PropertyObjectOperations. Every part contains the methods used to modify the identified object. For instance, ResourceClassOperations contains all the method that are used to change resource classes.

Finally the *matching* part of the resource manager takes the resource requests and matches them using the Jena SPARQL engine while taking into account the constraints and the resource and process rules that apply to the request. This is again implemented as an interface that makes it possible to change out the Jena SPARQL engine for another or even change the data model away from RDF. The functionality of the resource manager can be accessed either by the process interaction component or directly via an independent client. The client, seen in Figure 24, can be used for the management of both resource classes and resource objects, as well as for executing queries both using direct SPARQL input and a graphical user interface.

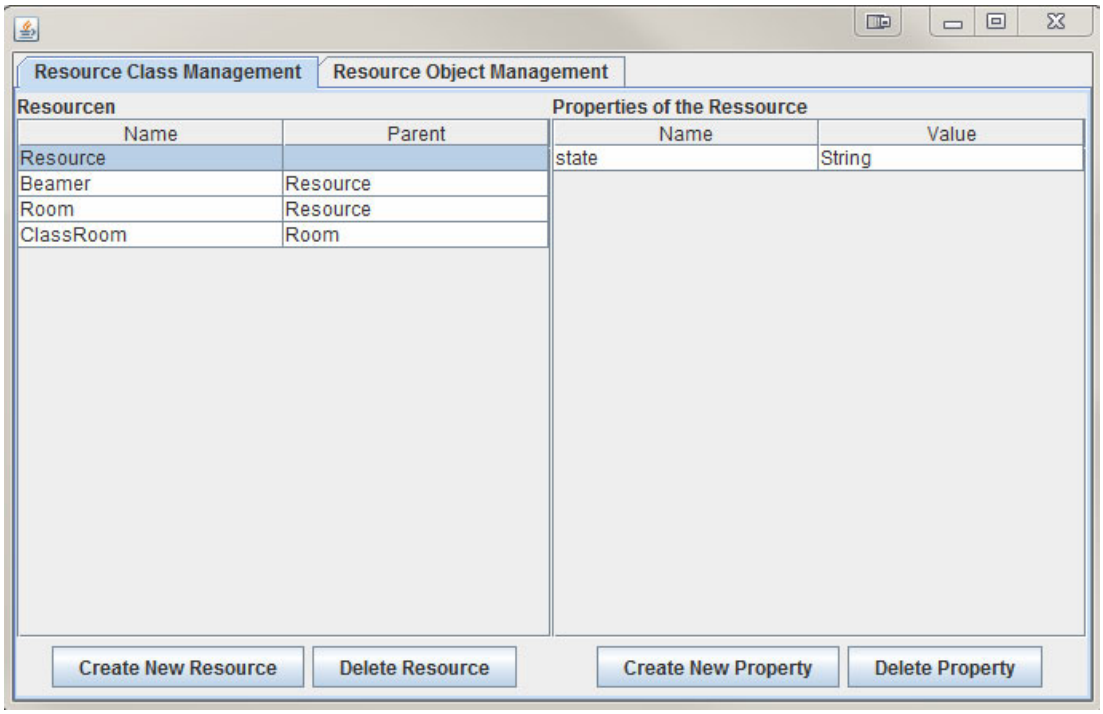


Figure 24, Resource Management Client

7.3.2 Process interaction component

The second part of the prototype handles the direct interaction with the AristaFlow Template Editor and works as an go-between between the business process management system and the resource manager. In contrast to the resource manager the process interaction component is highly dependent on the used business process management system and can not easily changed to work with anything but AristaFlow.

The process interaction component, the basic structure of which can be seen in Figure 25, is divided into two different parts, the *interaction model* and the *user interface*. The *interaction model* handles the objects that are created as representations of the resource requests, constraints and resource and process rules. It also establishes the connection to the resource manager and to the AristaFlow Template Editor itself. The interaction with the AristaFlow Template Editor is achieved via the AristaFlow Plug-In Data Container that allows the seperate interfaces for resource requests, constraints and resource and process rules to be integrated into the process template. The *user interface* makes use of the existing structure of the AristaFlow Template Editor and is implemented as an additional rider in the node properties.

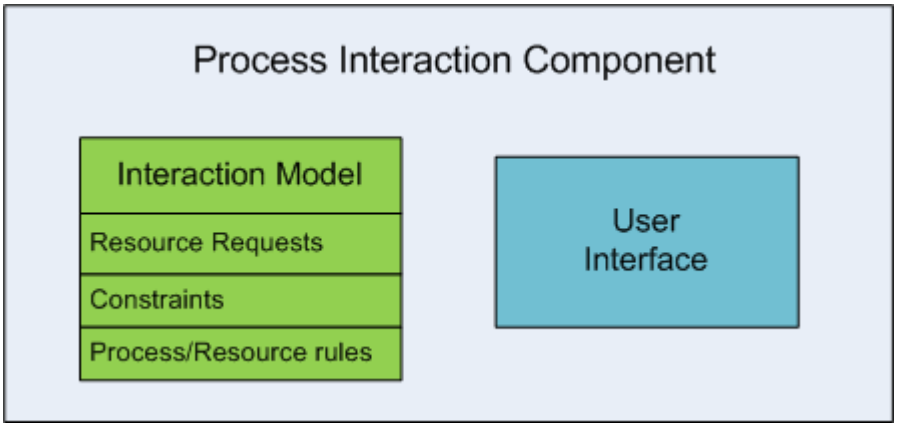


Figure 25, Process Interaction Component

By using this component, the modeler can add *resource request*, *constraints* and *process/resource rules* to any activity. For the resource request he selects a resource class out of a list of resource classes that are available and gives it a variable name. Then the SPARQL query is generated automatically and the user can input further restrictions in the FILTER clause of the query. When he adds a constraint, he selects two or more variables already established in the resource requests. He then can add restrictions to the variables so that, e.g., the first variable cannot be the same resource object as the second variable. In the same way he can decide that two variables have to be filled by the same resource object or

that, if one variable is filled by a specific resource object then another has to be filled by another specific resource object. When the user adds a process/resource rules, depending on the rule, a stub that contains the structure of the rule is displayed. The user can then select or input the variables that he wants. The design of the stub is, obviously, different, based on the rule in question. For the resource priority rule, it simply is an entry form for the number value of the priorities. In contrast, for the resource availability rule and the resource modification rule, it consists of IF/THEN statements that can be filled out by the user.

The resource requests, constraints and rules are all saved with the nodes. The user can change or delete them at any time.

7.4 Outlook

The prototype presented in this chapter is the first implementation of a resource management component for AristaFlow. It shows that it is possible to use an independent resource management component and combine it with an extension of the AristaFlow Template Editor to model requests, constraints and resource/process rules. Obviously this is only the beginning of an operational implementation of resource management in all phases of the business process lifecycle. Nevertheless, this prototype sets the foundation for the support of resource management both in process execution and process monitoring. Based on this implementation the further steps for a complete resource management extension for AristaFlow can be undertaken.

8 Summary

The aim of this thesis was to develop a model for the specification and management of complex resources in a process environment. For this, the requirements of such a model have been analyzed with respect to both the needs of the resources and the demands of business process management systems. A meta model for resources has been developed while taking into account already existing solutions and technologies.

The structure of the model has been presented in detail. It has been shown how the model was expanded, so that the resource management can be integrated into every part of the business process management system. The resource model has been tailored to the different phases of the process lifecycle.

During the process modeling phase, it can be selected which resources are needed for the execution of the different activities. The complex requirements and interactions between different activities have been met with the introduction of constraints and resource/process rules.

In the execution phase the reservation and allocation of resources have been integrated. A system for matching the resource demands to the available resources has been presented and mechanisms for the claiming and release of resources have been developed. It has also been shown, what role resources play in regards to adaptive process management.

The different aspects of monitoring in connection with resources have been presented in detail. It has been shown what resource information are logged and how, how this information is used both for post-execution analysis and real-time display and what role resources play in business intelligence.

Finally, an prototype of the model has been developed for the business process management system AristaFlow which uses the ADEPT2 model. There, it has been shown which components of the system can be implemented independently and which should be developed in direct contact with the AristaFlow Template Editor.

As shown in this work, the integration of specification and management of complex resources into a process environment is a viable and attractive way to produce a system which can manage resources efficiently. It can be expected that this subject will attract growing interest from the scientific as well as the business community.

Literature

[Aals04] W.M.P. van der Aalst: Workflow Mining: Discovering Process Models from Event Logs. In IEEE Transactions on Knowledge and Data Engineering, pp.1128-1142, 2004

[AFPS] AristaFlow Forum: Process Server, http://www.uni-ulm.de/einrichtungen/aristaflow-forum/aristaflow-bpm-suite/process-server.html (last visited 10.29.2012)

[AFPTE] AristaFlow Forum: Process Template Editor, http://www.uni-ulm.de/einrichtungen/aristaflow-forum/aristaflow-bpm-suite/process-template-editor.html (last visited 10.29.2012)

[AHW03] W.M.P. van der Aalst, A.H.M ter Hofstede, M. Weske: Business Process Management: A Survey. In *Lecture Notes in Computer Science, Volume 2678/2003*, pp. 1-12 2003

[ApaJen] Apache Jena. http://jena.apache.org/, (last visited 10.29.2012)

[APS10] W.M.P. van der Aalst, M. Pesic, M. Song: Beyond Process Mining: From the Past to Present and Future. In *Lecture Notes in Computer Science, Volume 6501/2010, pp. 38-52*, 2010

[BeCo11] T. Berners-Lee, D. Connolly: Notation3 (N3): A readable RDF syntax, http://www.w3.org/TeamSubmission/n3/, 2011 (last visited 10.29.2012)

[Buss98] C. Bussler: Workflow Instance Scheduling with Project Management Tools. In *Proceedings of the Ninth International Workshop on Database and Expert Systems Applications*, pp. 753-758 1998

[Clar05] K.G. Clark (Ed.): RDF Data Access Use Cases and Requirements. http://www.w3.org/TR/rdf-dawg-uc/, 2005 (last visited 10.29.2012)

[DAG+06] P. Dadam, H. Acker, K. Göser, M. Jurisch, U. Kreher, M. Lauer, S. Rinderle, M. Reichert: ADEPT2 - Ein adaptives Prozess-Management-System der nächsten Generation. In D. Spath, A. Weisbecker, O. Häss (Eds.): *Proceedings Stuttgarter Softwaretechnik Forum: Science meets Business - Aktuelle Trends aus der Softwaretechnikforschung*, 2006

[DAML] DAML: The DARPA Agent Markup Language Homepage. http://www.daml.org/ (last visited 10.29.2012)

[DaRe09] P. Dadam, M. Reichert: The ADEPT Project: A Decade of Research and Developement for Robust and Flexible Process Support - Challenges and Achievements. In *Computer Science - Research and Developement, 23 (2), pp. 81-97*, 2009

[ELD+96] D. Epema, M. Livny, R. van Dantzig, X. Evers, J. Pruyne: A Worldwide Flock of Condors: Load Sharing among Workstation Clusters. In *Journal on Future Generations of Computer Systems, 12*, pp. 53-65, 1996

[Halp01] T. Halpin: Object role modeling: An overview. white paper (online at www.ormfoundation.org), 2001

[LaSw99] O. Lassila, R.R. Swick: Resource Description Framework (RDF) Model and Syntax Specification. http://www.w3.org/TR/PR-rdf-syntax/, 1999 (last visited 10.29.2012)

[LiFo03] C. Liu, I. Foster: A Constraint Language Approach to Grid Resource Selection. In *Proceedings of the Twelfth IEEE International Symposium on High Performance Distributed Computing*, 2003

[MPP09] B. Motik, B. Parsia, P.F. Patel-Schneider (Eds.): OWL 2 Web Ontology Language XML Serialization. http://www.w3.org/TR/owl2-xml-serialization/, 2009 (last visited 10.29.2012)

[Mueh04] M. zur Muehlen: Workflow-based Process Controlling. Logos Verlag Berlin, 2004

[OWF+10] C. Ouyang, M.T. Wynn, C. Fidge, A.H.M. ter Hofstede, J.C. Kuhr: Modelling Complex Resource Requirements in Business Process Management Systems. In M. Rosemann; P. Green; F. Rohde (Eds.): *ACIS 2010 Proceedings*, Queensland University of Technology, Prisbane, 2010

[Pelt03] C. Peltz: Web Services Orchestration and Choreography. In *Computer, vol. 36/10*,pp. 46-52, 2003

[PMBK08] A Guide to the Project Management Body of Knowledge. Fourth Edition. Project Management Institute (PMI), 2008

[ProM] ProM: The leading Process Mining Toolkit, http://promtools.org/prom5/ (last visited 10.29.2012)

[PrSe08] E. Prud'hommeaux, A. Seaborne(Eds.): SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query/, 2008 (last visited 10.29.2012)

[RDF04] RDF Working Group (Eds.): RDF - Semantic Web Standards. http://www.w3.org/RDF/, 2004 (last visited 10.29.2012)

[Reic00] M. Reichert: Dynamische Ablaufänderungen in Workflow-Management-Systemen. PhD thesis, Uni Ulm, 2000

[RHEA04] N. Russell, A.H.M. ter Hofstede, D. Edmond, W.M.P. van der Aalst: Workflow Resource Patterns. BETA Working Paper Series, Eindhoven University of Technology, 2004

[RLS98] R. Raman, M. Livny, M. Solomon: Matchmaking: Distributed Resource Management for High Throughput Computing. In *Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing*, pp. 140-146, 1998

[RRKD05] M. Reichert, S. Rinderle, U. Kreher, P. Dadam: Adaptive Process Management with ADEPT2. In *Proceedings International Conference on Data Engineering*, 2005

[SarOxA] Wikipedia - Sarbanes-Oxley Act. http://en.wikipedia.org/wiki/Sarbanes-Oxley_Act (last visited 10.29.2012)

[SemWeb] Semantic Web - Main Page, http://semanticweb.org/wiki/Main_Page (last visited 10.29.2012)

[SWT] SWT: The Standard Widget Toolkit, http://www.eclipse.org/swt/ (last visited 10.29.2012)

[Tsan93] E. Tsang, Foundations of Constraint Satisfaction. Academic Press, 1993

[W3C09] W3C OWL Working Group (Eds.): OWL 2 Web Ontology Language Document Overview. http://www.w3.org/TR/owl2-overview/, 2009 (last visited 10.29.2012)

[Wesk07] M. Weske: Business Process Management: Concepts, Languages, Architectures. Springer, 2007

[WPI] Workflow Patterns Initiative: Workflow Patterns Homepage, www.workflowpatterns.com, (last visited 10.29.2012)

List of Figures

List of Tables

Name: Holger Bähren Matrikelnummer: 640038

Erklärung

Ich erkläre, dass ich die Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den . . . .

Holger Bähren