Resource Modelling in an Object-Oriented Process Modelling Language

Jürgen Jung
Bardo Fraunholz

IS Research Institute, University of Koblenz,
Campus Koblenz, Germany
{jjung|bardo}@uni-koblenz.de

Abstract

This paper introduces a modelling approach for requirements engineering at the initial and design phase of a project. In particular, it describes a preliminary framework for the design of resources in a modelling context and introduces a basic meta-model for the specification of resource types. One benefit of this approach is the provision of domain specific “packages” that can easily be used by domain experts but with the advantage to provide a structure that can simply be translated at the implementation stage, thus reducing the communication gap between domain and implementation experts.

Keywords:

Business Process Modelling, Requirements Engineering, Resources, Resource Modelling, Software Project

Introduction

Within software projects time spend on requirements has a fundamental influence on the success or failure as well as the budget of a project. Davis (1993) has summarised studies on the implications of mistakes at different stages during such projects and it appears that requirement errors are the most common and at the same time the most expensive to fix.

As a result it is important to understand what the requirements phase entails. Usually the analysis begins with the determination of the heart of the problem or the centre of the future desire. This means it is essential to understand the existing domain including the expectations of all existing stakeholders in order to move forward towards the desired domain.

In short this initial phase can be described by five steps (Leffingwell and Widrig 2000):

- Gaining an agreement on the problem definition.
- Understand the root cause – the problem behind the problem.
- Identify the stakeholders and users.
- Define the solution system boundary.
- Identify the constraints to be imposed on the solution.

Looking at requirements engineering with a focus on software development but from a business perspective, it is essential to understand how a business works and what it involves. Therefore, this paper describes how to model resources for business processes to assist the requirements engineering process with a vision on the possible generation of code.

Resources in Business Process Modelling

Resources are an essential prerequisite for the execution of business processes (Wöhe 2000). Starting with simple projects to complex workflow management systems, resources are at the centre of any description. Designing software requires us to develop descriptions of the relevant domain on an appropriate level of abstraction. Usually, it will be necessary to involve domain experts with a poor background in software engineering. Therefore, implementation level languages such as programming languages are not sufficient for describing systems during an early analysis phase.

However, natural language cannot be said to be adequate either. Natural language descriptions lack the precision and structure that is eventually required for implementation. Consequentially, conceptual models have been suggested to allow for "... descriptions of a world enterprise/slice of reality which correspond directly and naturally to our own conceptualisations of object of these descriptions." (Mylopoulus and Levesque 1985). While it is hard to say what makes a "natural" description, conceptual models promote an intuitive understanding by focusing on domain level concepts, by omitting implementation specific details and by featuring a graphical notation. A graphical notation is usually the best choice when communicating with domain experts, who are non computer scientists. At the same time some concepts of languages for conceptual modelling correspond to concepts of implementation level languages. A widely used language for conceptual modelling is the Entity Relationship Model (ERM). According to our experience, however, data models alone can hardly be said to be sufficient to document a system in a way that would advance discussions with domain experts and prospective users. Instead, we suggest a modelling approach covering various perspectives on an enterprise by a set of integrated models to be used from the starting point, the initial requirements statement, all the way to the final requirements document, to be passed on to the developers. Most requirements frameworks and languages are focused on the latter (e.g. Bubenko 1980; Greenspan et al. 1982; Du Bois 1995) and can therefore hardly be satisfactory and are consequently likely to miss the real needs of the users (Grudin 1988). Therefore we are working on an integrated modelling language enabling us to successfully undertake the early requirements process of understanding the domain. This is done in close collaboration with domain experts in order to avoid the primary risk of project failure (Curtis et al. 1988) and by focusing on existing business processes.

Curtis et al. (1992) identify three applications for modelling business processes:

- Business process reengineering
- Coordination technology
- Process-driven software development

Business process reengineering (BPR) aims at improving business processes within an organisation. The foundation for BPR is the identification and documentation of actual processes, to then analyse them with a focus on customer-orientation and process quality. Starting from the identified weaknesses, those processes are then rebuilt to meet customer demands and even improved to enhance customer satisfaction. *Coordination technology* mainly addresses workflow management systems (WfMS). A workflow management system's schema can be derived out of a business process model and serve as a coordination program for workflow management systems. However, the software-technical aspect of process models is not restricted to the – usually automated – generation of a workflow schema. A *process-driven software development* environment is based on the specification of the business processes within an organization. Such an environment supports the development of software derived from its models – including code generation.

Overview

In general, different levels of abstraction influence the conceptualization of a language. Aspects related to this are discussed in the following section **Levels of Abstraction**. Following this we present the basic conceptualization of a resource modelling language for an existing process modelling language. This presentation focuses on the modelling resource types as well as concrete resources. Accounting properties of resources and resource types are includes as well as the specification of the relationship between resources and types. Finally, this paper concludes with a summarising chapter and a look on future research.

Levels of Abstraction

Many everyday terms are ambiguous and can be interpreted accordingly (Frank and van Laak 2002). The term ‘resource’ might be interpreted as resource type (i.e. a resource class[1]) or equally likely as an instance of resource type (e.g. an existing database server in the corporate network). Furthermore, a resource class can be looked at from an intensional or an extensional point-of-view. The intensional interpretation of a class corresponds to a template for the specification of instances. A class named Person defines the attributes firstname, lastname and dateOfBirth for all persons. This template is used to instantiate new person-instances of the class Person. By contrast, the class extension is a set of homogeneous objects. Ambiguities of interpretations have to be avoided in the context of business process modelling for formal analysis, simulation or software development. In the same way, reuse of such concepts is restricted in domain or requirements engineering.

Furthermore, additional aspects of resource classification in a process modelling language have to be taken onto account. Such aspects are illustrated by example of a Personal Digital Assistant (PDA) in Figure 1[2]. Language features of a resource modelling language are specified by a meta-model. Instances of types in the meta-model are resource types such as the PDA in Figure 1. The resource type PDA may be classified from a technical or economic point of view. Technical aspects are model, processor speed, and size of the main memory. These aspects determine the restrictions of usage for such a resource type in a definite context. Economic properties of a PDA resource type address cost of operation and maintenance of a specific instance in a business process. Those aspects are described at different levels of abstraction by the same resource type.

An initialized resource type ‘Compaq iPaq H3660’ is an instance of the resource type PDA and can in turn also have instances (like an iPaq with the serial number 4775348 at the bottom of Figure 1). Consequently, we have a multi-level type-instance-relationship for the description of PDAs in a resource model. This relationship however is hard to handle, when using business process models as the basis for information systems development. Commonly object-oriented programming languages only offer a flat class-instance-relationship. Therefore, type-instance relationships of resource models can not be mapped directly to the programming language’s concepts.

To make things even more complex it is not clear whether special PDAs (e.g. a SmartPhone) are subtypes or instances of the PDA resource type. On the one hand, they may be regarded as subtypes because they have an additional features compared to standard PDAs. This feature is the build-in mobile telecommunication facility. On the other hand, a SmartPhone is of the

[1] The terms ‘type’ and ‘class’ are used synonymously within this paper.

[2] The notation used in this diagram is a variant of the entity-relationship-diagram notation.

resource type PDA because it might be a variant of an existing PDA with an additional mobile telecommunication facility (built-in or attached).

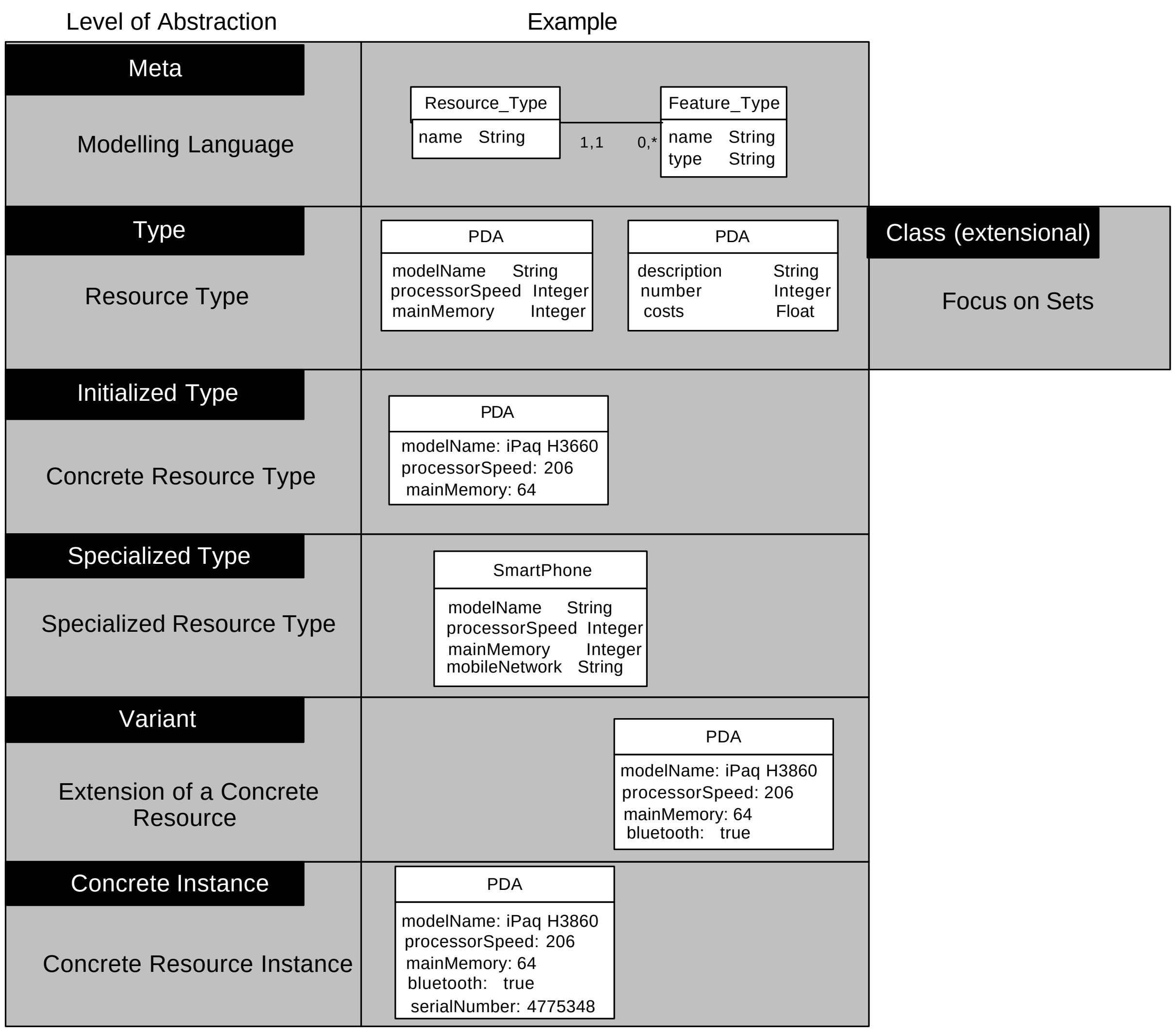

Figure 1: Levels of Abstraction

Conceptual modelling usually prescinds from concrete objects and changeable aspects. Hence, only types are allowed in conceptual models[3]. Nevertheless modelling of instances might be appropriate in certain situations:

- *The modelling of – anonymous – instances is used to formulate relationships between resources in different processes.* Such a relationship may be of the kind: The resource (instance) used in process A is also required in process B.
- *A prototypical instance in a process model reflects the average property-values of a resource type.* Costs for the usage of a resource are usually attached to the concrete resource instance and differ between instances. Hence, costs can not be assigned to resource types in conceptual modelling. But average costs of a certain resource type might be assigned to a prototypal instance which is used for formal analysis and simulation.

[3] Note that we do not discuss whether resource types or concrete resource types are appropriate in this context. This distinction is not important for the moment.

- *Concrete instances are required to describe currently used resources.* The first step of business process reengineering is the modelling of existing business processes. This model includes processes and resources as they are available at the moment. This might also include dedicated resources like special servers, relevant machinery, or technical engineers, which are hard to replace by other resources.

Hence, types and instances have to be modelled by a resource modelling language. This language should also allow different kinds of abstraction (technical and economic). Furthermore, different levels of abstraction have to be available with respect to type-instance-relationships. We will present our first approach for resource modelling in the following section. This approach will cover many of the discussed aspects.

Resource Specification in MEMO-OrgML

MEMO (Multiperspective Enterprise MOdelling) is a method for modelling organizations on different levels of abstraction and different perspectives. MEMO has been initiated by Frank (1994) and is the main research topic of the research group 'Enterprise Modelling' at the University of Koblenz (see e.g. Frank 1998 and Frank 1999). MEMO includes several languages for modelling static, functional and dynamic aspects of an Enterprise. One of these languages is the MEMO-OrgML (Organization Modelling Language), which supports modelling of organizational structures and processes (An earlier version on process modelling see Wenzel 1997; formal specification is discussed in Zickhardt 1999). Resource modelling has not been part of the first conceptualization of the MEMO-OrgML. Our first approach for modelling resources with MEMO-OrgML is presented here.

We have identified two ways for the provision of resources in a business process modelling language:

- Specification of a dedicated resource modelling language for the description of resources and resource types by the user of a modelling language
- Offering a fixed number of predefined resources and resource types to be used in a process model

The first approach addresses a flexible language for the specification of any resource or resource type. A user might define resource types as needed within specific processes. Despite the flexibility of such a modelling language, this approach lacks ease-of-use. Every new resource type has to be specified and the user has to learn a specific language for the specification of resource types. Using predefined resources types on a user's level – i.e. types like database server, service employee, truck, production plant, and so on – might be easier to handle by a domain expert than abstract concepts of a general resource modelling language. In consequence, the provision of predefined resource types solves this problem, but restricts the user to a fixed set of available resource types. He might easily (re-) use them in his models but it lacks possibilities for the addition of missing resource types.

Hence, we decide to offer both within the MEMO-OrgML: a dedicated resource modelling language and a number of predefined resource types sets for specific purposes. Predefined resources types are specified by the resource modelling language. Obviously, it is hardly possible to provide resource types for all possible needs. Therefore we restrict the development of resource types to some domains. All resource types according to a domain are grouped in so called packages[4]. We present the conceptualization of some aspects of our resource modelling language by its meta-model in the following paragraphs in sections **Basic Meta-Model of the Resource Specification Framework**, **Meta-Model of Economic Resource Features**, and **Meta-Model of Information-Resources**. Basic ideas for the provision of concrete resource types are introduced in section **Concrete Resource Types**.

Basic Meta-Model of the Resource Specification Framework

The foundation of the resource modelling language is the meta-model for the specification of resources and their types. The current version of the core of this language is presented in Figure 2. The perspective of this meta-model is the description of abstract resources on a conceptual level. Resources are seen from the perspective of a domain expert, who deals with existing resource types assigned to business processes. Accounting aspects are excluded and will be discussed in the following section. The core resource modelling language aims to offer domain-specific resource types like machinery, raw material, staff qualification, and their respective types.

At the top level of the class hierarchy of the meta-mode we distinguish between compound and elementary resources types. A compound resource type is an abstract resource type, which is composed by other abstract resource types. Hence, a compound resource type may consist of several elementary or compound resource types. Elementary resource types are specialised to human, physical, and intangible resource types. A human resource type corresponds to an organisational unit or a role filled by an employee (van der Aalst and Hee (2002) discuss the correspondence of – in general human - resources to roles and organisational units). Physical resource types comprise all tangible objects used within a business process. Subtypes of physical resource types are resource types, which

- are used for the completion of a process (OperationalResourceType),
- are consumed by the completion of a process (ConsumptionableResourceType), or
- store information (MediumType).

[4] The concept of a package and resources or resource types is different from UML packages. The term package might be ambiguous when using UML diagrams for the description of a meta-model. Within this paper, the meaning of the term package only refers to sets of resources or resource types.

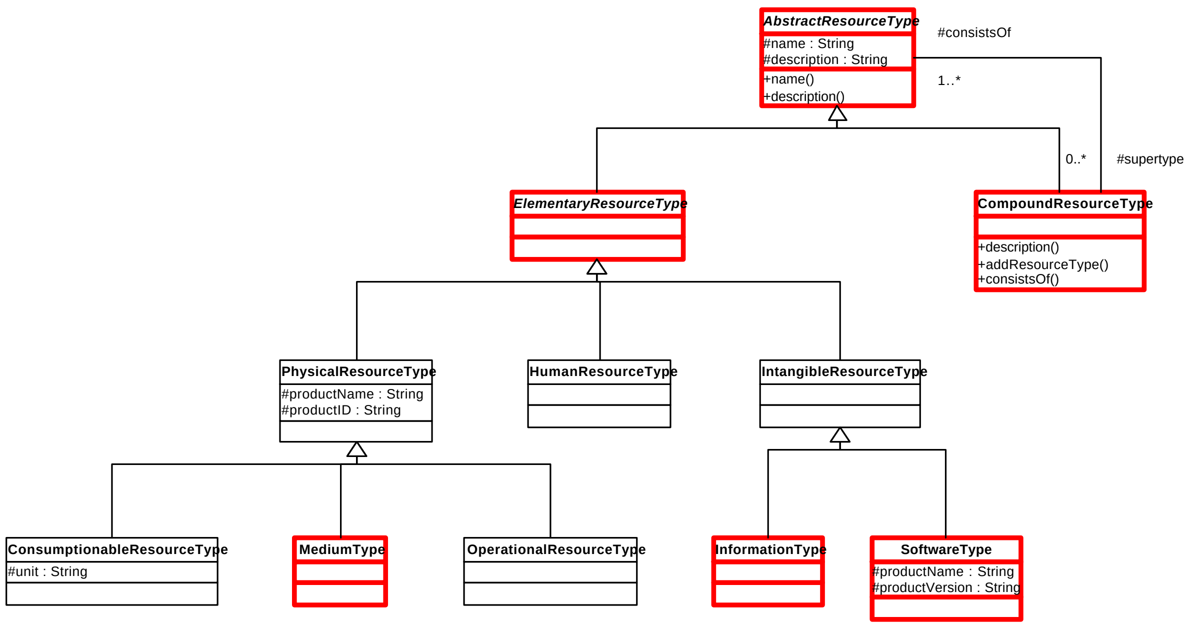

Figure 2: Basic Resource Types[5]

Operational resource types are physical resource types, which are used within a process and are still available after its completion. Examples are machinery, tools, and vehicles. They are all used for processing but remain available. In contrast, consumable resource types are a prerequisite for a process and are transformed during the execution of a process. Raw and operational material as well as spare parts are used by a business process and are transformed to a (partial) product. They (individually) will not be available for other processes. Information containing media does not fit the differentiation between operational and consumable resource type. A medium or its information might be out-of-date after a process or it might be a prerequisite for subsequent processes. Intangible resource types represent all resource types, which are neither physical nor human. Two subtypes have been identified up to now - information and software[6]. These can not exist on their own, because they need a medium for representation – persistent or for transmission. Information resource types and medium types will be discussed in section **Meta-Model of Information-Resources**.

[5] Thick outlined class symbols have the same semantics as any simple outlined class symbol in this paper. The different outline originates from the modelling tool, which emphasises class symbols used in several class diagrams.

[6] It is not elaborated at the moment, whether there are additional intangible resource types or if software and information share a specialisation-generalisation-relationship.

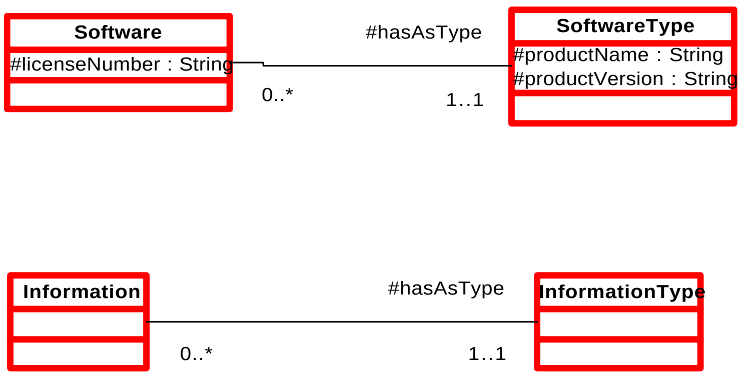

Figure 3: Resources and their corresponding types

An existing resource represents a dedicated instance of a specific resource type. A truck with a certain numberplate is a vehicle resource of a truck type. A database server instance corresponding to a dedicated computer, running a specific database management system, assigned to a concrete administrator, is a resource of a database-server type. The relationship between resources (instances) and their corresponding types is established by a hasAsType-relationship. Examples for intangible resources are given in Figure 3. An installed program with an assigned license number is related to a software type, represented by the name and version of the software package by the hasAsType-relation. Also information such as the instance of an invoice is related to the invoice type.

Meta-Model of Economic Resource Features

From an economic point-of-view, resources are usually associated with costs of usage within a business process. The way in which resources are allocated to tasks is very important to the efficiency of a workflow (van der Aalst and Hee 2002). Hence, a resource modelling language has to consider economic aspects of the resource usage. Regarding different business process and resource types, costs can be determined differently. A resource can at any given time be used within several processes or just one. If it is shared between processes its availability is restricted by a maximal capacity; represented by a capacity unit and an upper bound of numbers of that unit. The usage of such resources in different processes has to be specified by the amount of units required for that specific process. A database server is an example for a shared resource. The work load of a database server is often specified by a maximum number of simultaneous transactions. Thus, the capacity unit is 'transaction' and the maximal capacity is the highest number of transactions at a given point of time.

Costs of exclusively used resources can be interpreted differently – depending on the kind of resource use. Resources are accounted for on a fixed price per unit or on a time base. Another possibility is the assignment of abrasion or depreciation for the use of a resource. Fixed prices usually form the calculation base for spare parts as well as operational and raw material. Some staff and machinery may also be calculated for by fixed price. Additionally, those resource types are characterized by their abrasion or depreciation, while used for a process. Hence, the kind of accounting is orthogonal to the type of a resource.

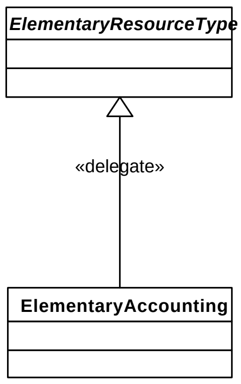

Figure 4: Realisation of Accounting

As stated in the chapter on **Levels of Abstraction**, resources can be interpreted from different viewpoints. The basic framework for the specification of resources and resource types follows a conceptual point-of-view and economic features which we deferred for the moment. The core problem lies in the different classification of resources. Two major typing schemes discussed in this paper are the conceptual and the economic classification. There are other kinds of classification like availability, quality, or responsibility. Hence, we should consider a multiple classification of resources and resource types. Additionally, the economic classification of a resource might change in time or between different business processes. A fixed-price defined resource for one process might be accounted on a time base in another. In consequence, the classification from an economic point-of-view can be dynamic (see Odell 2002 for discussion on multiple and dynamic classification).

Different perspectives for the classification of resources in the resource modelling language are realized by delegation (as presented in Frank and Halter 1997). The primary classification of resources is based on a domain expert perspective as presented in **Basic Meta-Model of the Resource Specification Framework**. Other perspectives – such as accounting – are attached to resources by delegation (see Figure 4). Delegation means that a resource might satisfy a certain perspective by playing a special role. The role of an elementary resource type (ElementaryResourceType in Figure 4) is represented by an instance of the class ElementaryResourceType. This elementary accounting encapsulates all economic properties of a resource and replaces the resource in every economic context. All non-economic aspects of a resource within an economic context are represented by the resource (type) itself. Hence, the elementary accounting represents a resource whenever accounting properties are requested and delegates all other inquiries to the resource itself.

Meta-Model of Information-Resources

Information types correspond to some kind of knowledge of an organisation. Information types might be dedicated document types like order, assembly list, and invoice. Furthermore, an information type reflects guidelines, regulations, and directions for the execution of business processes. All these kinds of information are covered by the meta-model in Figure 5.

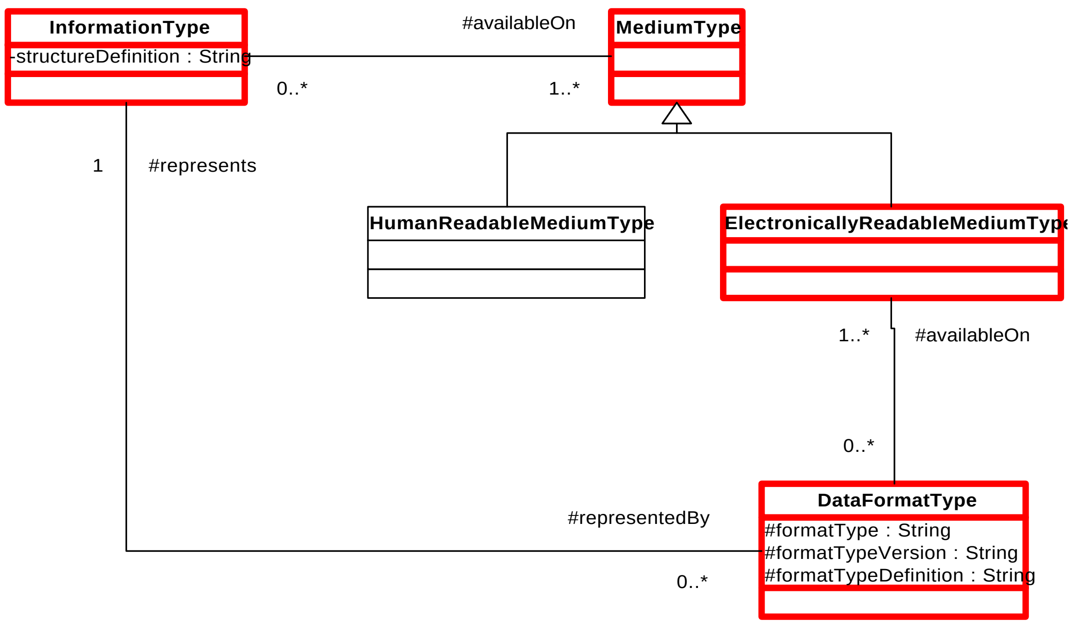

Figure 5: Information Type

An information type is defined by its structure, as described by the following examples:

- An invoice consists of a date, sender, receiver, and several positions.
- A guideline is structured by its corresponding document, the corresponding applicability, and its contents.

This structure is encapsulated by the attribute structureDefinition. This definition is independent of a special data definition. All information might be structured by a formal data definition but need not necessarily be. Consequently, the structure definition of an information type is independent of a formal data definition. Such a data definition might be an XML-DTD, an SQL table definition or another structure definition language. If there is a formal structure definition, it might be assigned to the information type. Additionally, information is represented on a medium. A medium is a carrier for information. We distinguish between human and electronically readable media. A human readable medium corresponds to a representation for human readers, like a paper-based hardcopy or a screen output. Electronically readable media are those kinds of media, which allow a computerised processing of information. Storage of information on an electronically readable medium is only allowed if a data-format for this information is specified.

Concrete Resource Types

In addition to the specification of a general resource modelling language, we also define concrete resource types based on this langue. Taking into account the impossibility of specification of all existing resources, we will concentrate on the provision of resource types for some selected domains. Resources of a specific domain are grouped by a package, a set of most needed resources in a domain. Examples for such packages are a set of resources for logistical processes or a set of IT-resources. The following table presents a few examples:

Package	Resource Type
Logistics	Dispatcher
	Vehicle
	Driver
	Transportation Order
	…
IT	Administrator
	Server
	Database
	Application-Server
	Database-Server
	…

Table 1: Examples of resources for the logistical and IT-Packages

Obviously packages are not disjoint. The modelling of resources for logistical processes requires resources of the logistics domain and IT-resources. Logistical resources are used for the execution of processes and IT-resources are involved for planning and administration of these processes. Hence, modelling of resources for business processes requires several packages. From a user’s point of view, modelling of resources within business processes requires the selection of

- appropriate packages and
- necessary resource types.

Using available resource types does not incriminate the task of specification of these resource types to the user of a process modelling language. One might just choose between resources on a well-known level of abstraction. Domain expert will know about the concepts of their domain but they will not know specifics of modelling languages like a dedicated resource modelling language. Someone else might just use pre-defined resources, which are already specified by the resource modelling language. The relationship between a concrete resource type and its formal specification is given in the following Figure 6.

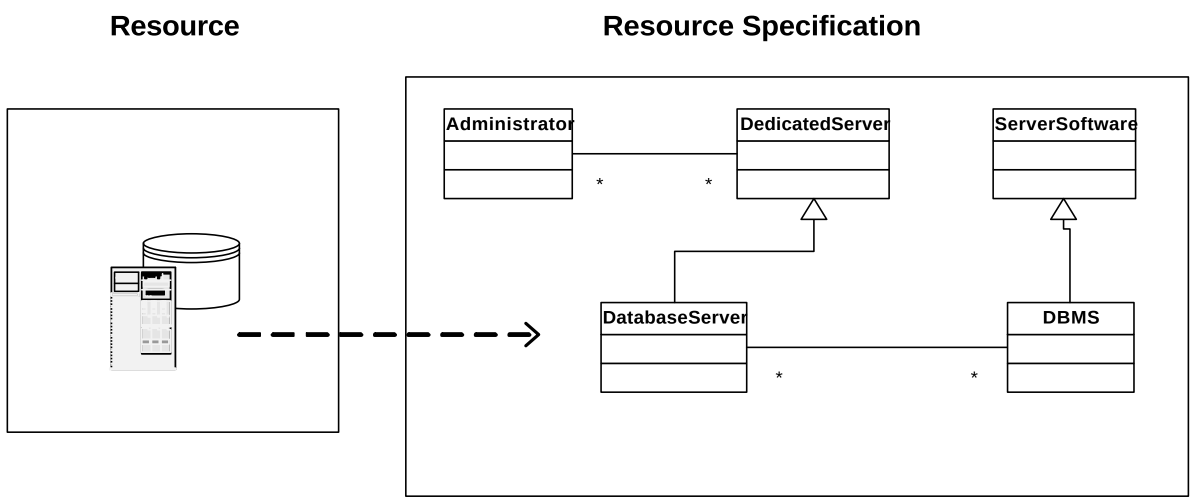

Figure 6: A concrete resource and its Specification

A resource type 'database server' is provided by the IT-package (left side of Figure 6). The specification of the database server has already been made by the creator of this package (right side of Figure 6). The implementation of the packages has not been tackled. This is work in progress and advances will be presented at a later stage.

Conclusion and Future Work

This paper presented our first approach for modelling resources in MEMO. By showing the relevance of resources for business process modelling and requirements engineering as well as also pointing out some of the challenges of resource modelling, we developed a preliminary framework for the specification of any resources and resource types as well as several sets of predefined resource types. Also some extracts from the meta-model of the resource specification language were presented. However, at this stage we only focussed on the basic meta-model for the specification of resource types, the modelling of information types and media, and the modelling of economic properties of resources. The need for adapted sets of resource types in different domains has also been introduced. This is especially important for the business context, so that we can provide adequate support at all levels of business and to all stakeholders. We aim for a modelling approach providing a multi perspective view of on business. Those adapted sets – called packages – support a domain expert on reusing predefined resource types within his specific context.

Because of the early stage of our research, the development of the resource modelling language; its quality; the applicability in business process modelling; and user acceptance has not been evaluated greater detail. Further study has to be done on the verification of the meta-model of the language. Existing concepts have to be proven in different contexts and missing concepts have to be identified. Hence, the quality of the resource modelling language depends on a broader application of this language. An important context for the evaluation of the resource modelling language is the development of resource type packages for different domains. These packages will provide the user with domain specific resources adopted for the user in a specific context. Examples such as logistical resources, IT-resources, and resources in the context of project management will be developed in the near future. Knowledge and experience evolving from the realization of these packages will directly affect the conceptualization of the resource modelling language.

References

van der Aalst W. and Hee K. (2002): Workflow Management – Models, Methods, and Systems, MIT Press, Boston.

Du Bois Ph. (1995): *The Albert II Language – On the Design and the Use of a Formal Specification Language for Requirements Analysis*, Ph.D. Thesis, Department of Computer Science, University of Namur.

Bubenko J. A., (1980): *Information Modeling in the Context of System Development*, Proc. IFIP, Elsevier, North-Holland, pp. 395-411.

Curtis B., Krasner H. and Iscoe N. (1988): *A Field Study of the Software Design Process for Large Systems*, Communications of the ACM, 31(11), pp. 1268-1287.

Curtis B., Kellner M. I. and Over J. (1992): *Process Modeling*, Communications of the ACM, 35(9), pp. 75-90.

Davis, A. (1993): Software Requirements: Objects, Functions, and States, Prentice-Hall, Englewood Cliffs, NJ.

Frank U. (1994): *Multiperspektivische Unternehmensmodellierung – Theoretischer Hintergrund und Entwurf einer objektorientierten Entwicklungsumgebujng*, Oldenbourg.

Frank U. (1998): *The MEMO Meta-Metamodel*, Research Report of the Institute for IS Research, University of Koblenz, No. 9.

Frank U. (1999): *MEMO: Visual Languages for Enterprise Modelling*, Research Report of the Institute for IS Research, University of Koblenz, No. 18.

Frank U. and Halter S. (1997): *Enhancing Object-Oriented Software Development with Delegation*, Research Report of the Institute for IS Research, University of Koblenz, No. 2.

Frank, U. and van Laak B. (2002): *Anforderungen an Sprachen zur Geschäftsprozessmodellierung*, to be published as a Research Report of the Institute for IS Research, University of Koblenz.

Greenspan S. J., Mylopoulos J. and Borgida A. (1982): *Capturing More World Knowledge in the Requirements Specification*, Proc. Int. Conf. on Software Eng., Tokyo.

Grudin J. (1988): *Why CSCW Applications Fail - Problems in the Design and Evaluation of Organizational Interfaces*, Proc. Conference on Computer-Supported Cooperative Work, ACM Press, New York, pp. 85-93.

Leffingwell, D. and Widrig D. (2000): *Managing Software Requirements – A Unified Approach*, Addison-Wesley, Boston, MA.

Mylopoulus J. and Levesque H.L. (1985): *An Overview of Knowledge Representation* in Brodie M. L., Mylopoulos J., Schmidt J. (Eds.) On Conceptual Modelling. Perspectives from Artificial Intelligence, Databases and Programming, Springer, Hamburg, pp 3-17.

Odell J. J. (1992): Dynamic and multiple classification, *Journal on Object-Oriented Programming*, January 1992, pp. 45-48.

Wenzel J. (1997): *Entwurf einer Modellierungssprache zur Beschreibung von Geschäftsprozessen im Rahmen der Unternehmensmodellierung*, Diploma Thesis, University of Koblenz.

Wöhe G. (2000) *Einführung in die Allgemeine Betriebswirtschaftslehre*, 20th Edition, Vahlen, München.

Zickhardt J. (1999): *Integrierte Syntax und Semantik einer Objektmodell- und einer Geschäftsprozeßsprache: Eine EER/GRAL- Formalisierung und Semantikbeschreibung in Z der MEMO-Komponenten OML und PML*, Diploma Thesis, University of Koblenz.