A Usage-Based Unified Resource Model

Yves Wautelet,
HUBrussel / KUL University of Leuven

Samedi Heng,
Manuel Kolp,
Louvain School of Management
A Usage-Based Unified Resource Model

Yves Wautelet, HUBrussel / KUL University of Leuven
Samedi Heng, Louvain School of Management
Manuel Kolp, Louvain School of Management

Summary

Most of the time engineering methodologies focus on the modeling of functional and non-functional requirements of the system-to-be with no or poor representation of resource elements. Resources are nevertheless central in most industrial domains and do have an important impact onto the performance and feasibility of software requirements. That is why we propose, in this paper, an ontology for resource representation centered on its Usage i.e., the concepts of functionality for Resource Objects and competency for Resource Agents. This ontology does not tackle the particular problem of service level agreement which is a complementary dimension but rather focuses on how resources can be represented and handled at runtime. Heterogeneous resources can thus be represented in a unified manner within the context of “resource-intensive” domains where information systems are developed. Moreover, it could also be used to develop a specific heterogeneous monitoring system with, for instance, the agent technology so that it acts for interoperability purposes. The ontology proposal is applied on a case study in the industrial context of a steel industry, namely CARSID where lots of resources are collaborating to achieve defined services. The purpose is to show the applicability of the ontology in an industrial context where resources play a central role for the information system.

Keywords: Ontology, Resource Model, Software engineering, Information System.

This work was presented at the 24th International Conference on Software Engineering and Knowledge Engineering (SEKE), July 2012.

Corresponding author:
Mr. Samedi Heng
Center for Excellence PRISME
Louvain School of Management / Campus Louvain-La-Neuve
Université catholique de Louvain
Place des Doyen 1
B-1348 Louvain-La-Neuve, BELGIUM
Email: samedi.heng@uclouvain.be

The papers in the WP series have undergone only limited review and may be updated, corrected or withdrawn without changing numbering. Please contact the corresponding author directly for any comments or questions regarding the paper.
President-ILSM@uclouvain.be, ILSM, UCL, 1 Place des Doyens, B-1348 Louvain-la-Neuve, BELGIUM
A Usage-Based Unified Resource Model

Yves Wautelet$^1$, Samedi Heng$^2$ and Manuel Kolp$^2$

January 2014

Abstract

Most of the time engineering methodologies focus on the modeling of functional and non-functional requirements of the system-to-be with no or poor representation of resource elements. Resources are nevertheless central in most industrial domains and do have an important impact onto the performance and feasibility of software requirements. That is why we propose, in this paper, an ontology for resource representation centered on its Usage i.e., the concepts of functionality for Resource Objects and competency for Resource Agents. This ontology does not tackle the particular problem of service level agreement which is a complementary dimension but rather focuses on how resources can be represented and handled at runtime. Heterogeneous resources can thus be represented in a unified manner within the context of “resource-intensive” domains where information systems are developed. Moreover, it could also be used to develop a specific heterogeneous monitoring system with, for instance, the agent technology so that it acts for interoperability purposes. The ontology proposal is applied on a case study in the industrial context of a steel industry, namely CARSID where lots of resources are collaborating to achieve defined services. The purpose is to show the applicability of the ontology in an industrial context where resources play a central role for the information system.

Keywords: Ontology, Resource Model, Software Engineering, Information System.

$^1$HUBrussel/KUL University of Leuven, Stormstraat, 2, 1000 Brussels, Belgium. E-mail: yves.wautelet@kuleuven.be.

$^2$Louvain School of Management, Université catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium. E-mail: samedi.heng@uclouvain.be, manuel.kolp@uclouvain.be.
1 Introduction

Poor focus on detailed representation of complex resources within software development methodologies results omitting taking into account important aspects of the software problem that could be represented using a common pattern to ease the work of both the software analyst and designer. We consequently propose in this paper a common ontology allowing to model each type of resource in a unified and standardized manner, i.e. independent of their type, domain and interface. Indeed, resources are useless when they do not provide an added value for a functional or operational execution so that the ontology is driven by the Resource Usage i.e., functionalities for Resource Objects and competencies for Resource Agents. The ontology proposes to centralize resources’ offer and demand through their Usages. More specifically, we present an ontology for resource representation and handling that can be used as a general reference into resource intensive domain where information systems are developed. The contribution is located at design stage so that resources identified during analysis can be mapped into functionality and competency providers at runtime for dynamic resource allocation and reservation. In that perspective, a generic multi-agent system for resource-monitoring could be easily build up on its basis. This, within a functional software application, furnishes an intelligent – i.e., optimized on the basis of the available information – resource allocation mechanism. This ontology represents a contribution at design stage but is intended to be used in the context of a larger development methodology [14, 13] within the engineering of software services. Further developments as well as advanced strategies for resource monitoring as proposed in [6] are outside the scope of the paper and are left for future perspectives.

The rest of the paper is structured as follows. Section 2 discusses the structural choices of the ontology notably through the concepts of functionality and competency; the distinction between resources and actors and, finally, resource composition and hierarchy. Section 3 overviews the conceptual model itself while Section 4 is devoted to a case study in the steel making industrial environment of CARSID. Finally, Section 5 briefly depicts related work and Section 6 concludes the paper.

2 Modeling Resources

This section discusses the concepts of functionality and competency, distinguishes resources from actors and discusses resource composition and hier-
archy.

2.1 Modeling Resources with their Use

There are numerous definitions of the concept of resource. The Merriam-Webster dictionary refers it as a person, asset, material, or capital which can be used to accomplish a goal. While remaining basic, this definition has the interest of identifying the fact that a resource is not a process (something functional/operational) but should be used for the proper achievement (the goal) of such process through action(s). This consequently leads to the intuition that resources do own particular skills that are required in a functional context (for the realization of processes or services). Within a pool of resources, the requesting service is thus only interested by resources helping him achieving its functional requirements. Nevertheless, defining their type, resources are inanimate and do have a functional utility (this is the case of Resource Objects) or are able to behave (this is the case for Resource Agents). Resources’ offer and demand can consequently not be centralized into a single concept so that Resource Objects are functionality providers while Resource Agents are competency providers. Resources offer functionalities and competencies while services demand them.

A functionality is the ability of a resource to be used for achieving a specific purpose. In this sense the resource owning the functionality is passively used during a process execution so that it is specific to Resource Objects. Moreover, the concept of competency used in human capital and business resources management allows stakeholders and stockholders to reach resources unification and integration to optimize the capacity needed to achieve the vision, mission and objectives of their organization.

Following [5], a competency is an aptitude to know, to know how and to behave while [9] defines competencies as statements that someone, and more generally some resource, can demonstrate the application of a generic skill to some knowledge, with a certain degree of performance. In other words, competencies are observable skills, knowledge or know-how defined in terms of behaviors and processes required for performing job and business activities in a successful way. They are consequently owned by resources able to behave i.e. Resource Agents (see section 2 for a complete justification and definition). Knowledge and learning are specific to human/organizational resources and artificial intelligence systems. Behavior can be considered as the ability owned by Resource Agents, making them useful in an operational context.

In the context of the proposed ontology, resources are thus components
encapsulating functionalities and competencies. What fundamentally distinguishes those two concepts is that:

- a functionality is transparent in the sense that the environment is perfectly aware of what the Resource Object can deliver at defined quality level. The functional offer of Resource Objects can be said to be stable since they cannot acquire more functionalities - or lose existing functionalities - unless being transformed (or degraded);

- a competency is partially hidden since initial evaluation is necessary to assess the quality level at which a Resource Agent can deliver it; it is also evolutionary since it can be delivered at higher quality level later into the system life cycle thanks to the inherent ability to learn from Resource Agents.

Resources can nevertheless not always lead to an absolute and successful achievement of the functionalities or competencies they advertise so that a quality level should be associated to each of them. Competencies are indeed not neutral and are furnished following a quality level so that without being perfectly rational, their realization is non-deterministic and must be estimated. The probabilistic measure is, in our model, encapsulated in the concept of quality level; we indeed consider the probability as being an aspect of the quality ontology presented hereby. A Resource Agent providing a competency with a lower realization rate will be attributed a lower quality level following the particular quality ontology. Quality ontologies are domain specific and such a characterization is positioned into the proposed resource ontology but not instantiated (see Section 3). Similarly, furnishing competencies sometimes requires resource composition – i.e., the use of a defined set of resources configured in a defined way – so that resource hierarchization is required (as detailed in Section 3.3). Quality ontology proposal we can link this work to are provided in [7].

Moreover, competencies can also be acquired in the case of an individual or an organization through formal or non-formal procedures. This suggests Resource Agents evaluation through a formal process. For example, an ISO accreditation is, for an organization (which is here a Resource Agent) receiving it, a certification to have competencies to furnish a product or a service at some defined minimal quality level. In the same way, the follow-up and successful achievement of a study program in a given school or university is a process for a human (which is here a Resource Agent) to acquire and accreditate (through the obtention of a diploma) defined competencies at a
specified quality level. The processes of acquisition and assessment of competencies within particular (or a particular set of) resources is outside the scope of this paper. Functionality and competency ontologies are domain specific and such a characterization is positioned into the proposed resource ontology but not instantiated (see Section 3). Specific competency ontology proposals we can link this work to are provided into [5, 9].

2.2 Resources, Actors and Services

In high level analysis formalisms such as the i* modeling framework ([15]), human or machine resources can be modeled as actors depending on each other involved in the achievement services; this vision assumes an intentional dimension and two parties, the service consumer and the service provider. With respect to the service ontology presented in [3], this is the Service Commitment level (prescriptive level) but the resource ontology we propose in this paper is (also with respect to [3]) at Service Process (design and implementation levels). Following [4], Services are “high-level” elements i.e., coarse-grained granules of information that encapsulate an entire or a set of business processes. That is consequently the level where service consumers and service providers are specified and, since the proposed ontology is at service process level we do not specify them into the contractual aspects of resource reservation and use (see section 3.2).

3 An Ontology for Resource Representation

We define in this section a conceptual model for resource representation. In this perspective, Figure 1 depicts the relevant concepts and their dependencies using a class model [8].

3.1 Basic Concepts

Services require, for their proper execution (or realization), a series of Functionalities and Competencies at a defined QualityLevel furnished by one or more Resources. The use of a Resource by a Service is only conceivable by setting up a Contract for a defined period of time and defined QualityLevel and Cost. All of the system Resources are kept in the ResourceList while their “real-time” availability is given through their Status. Those concepts are formally defined hereafter.

A tuple $\left\{ (u_{t}, q_{r_{u_{t}}}), \ldots, (u_{t+m}, q_{r_{u_{t}},m}) \right\}, Res^r$ is a Resource $r$, where $u_{t}$ is a Usage. A resource furnishes Usages (which can be Functionalities or
Figure 1: An Ontology for Resource Representation.

**Competencies** required by **Services** to contribute to their fulfilment) at **Quality Level** $q^r_{us}$, ($q^r_{us}$ follows a particular quality ontology). $Res^r$ is assumed to contain all additional properties of the resource not relevant for the present discussion, yet necessary when implementing the solution. Resources belong to the set $RL$ ($RL$ stands for **Resource List**).

More precisely, resources deliver **Functionality**s if they are **Resource Objects** or **Capabilities** if they are **Resource Agents** (see Section 2 for a complete justification). A **Usage** is a generalization of **Functionality** and **Competency** and is compulsorily one or the other. That concept allows to tackle the problem in a generic manner. The reader should note that the distinction between **Resource Agent** and **Resource Object** is not in the sense of agent or object-orientation but in the sense of **Resource Objects** being inanimate; so even if the program is object-oriented, it is still classified as **Software Agent**, since what matters is modeling behavior.

The model is independent of any **Functionality** and **Competency** ontolo-
gies; it can for example be used through the competency ontologies defined in [5, 9]. The FunctionalityParameter and CompetencyParameter classes are assumed to contain all of the dimensions of these custom ontologies.

As previously discussed, Services are considered here as coarse-grained granules of information that encapsulate an entire or a set of business processes. With respect to the ontology proposed in [3], services are thus viewed at the level of Service Processes - i.e., design and implementation levels so that services' format and content depend on the implementation paradigm being used (procedural, object, agent, etc.). For that purpose, they are not further specified (e.g., in terms of pre/post condition and invariants) and defined here. Indeed, for example, within the i* framework, a Service can be a Goal or Task (while functionalities and competencies are at Capability level); in UML/J2EE technology [1] it can be a Use Case (while functionalities and competencies are at Method Call level). Further functional considerations of the Service are outside the scope of this ontology which tackles a lower level of abstraction (granularity).

The set of resources $RL$ (ResourceList) can be used as yellow pages referring the resources present into the system; their status – i.e., information about their availability at a given moment of time – must then be evaluated using the characterization that follows.

A ResourceStatus, $status_i$, is $\langle status_{i}^{pre}, \tau_i, status_{i}^{post} \rangle$, where $status_{i}^{pre}$ describes the preconditions for establishing a contract (following a particular ResourceStatus ontology), $\tau_i$ is a specification (following a particular API) on how the resource can be interfaced with and $status_{i}^{post}$ describes the postconditions to properly end up the contract (following the same ResourceStatus ontology).

The ResourceStatus ontology can be customely implemented within the model. The StatusParameter class is assumed to contain all of the dimensions of this custom ontology. A Configuration is a subset of ResourceStatus elements; this concept can be used to predefine quality levels before runtime for optimization purpose in the form of a “caching” system.

Quality and cost ontologies can be implemented within the model on a case by case basis. The QualityParameter and CostParameter classes are assumed to contain all of the dimensions of these custom ontologies.

### 3.2 Contractual Aspects

As previously evoked, the resource ontology assumes a “higher-level” (service) dimension. A resource in use within the realization of a defined service can, for a time shift be not available and has to be invoiced to the consum-
ing process (or its cost center). Consequently such a transaction has to be tracked by a formal element managing the resource reservation and invoicing aspects: the contract. Finally, the use of a resource has a cost. Cost ontologies are domain specific and such a characterization is positioned here without being instantiated. A specific cost ontology proposal we can link this work to is provided into [11].

A Contract $cont_i$ is $\langle Serv^s, us_i, res_{i}^{Qual}, res_{i}^{Cost}$, $res_{i}^{BeginTime}, res_{i}^{EndTime} \rangle$ associates a Service $s$ requiring the Usage $us_i$ (Functionality or Competency) provided by the selected Resource $res_i$, where:

- $res_{i}^{Qual}$ specifies the minimal ensured quality level. Its definition follows a particular quality ontology. Whatever the specific quality ontology, expected qualities are likely to be specified as (at least) $res_{i}^{Qual} = \langle (p_1, d_1, v_1, u_1), \ldots, (p_r, d_r, v_r, u_r) \rangle$, where:
  - $p_k$ is the name of the QualityParameter;
  - $d_k$ gives the type of the parameter (e.g., nominal, ordinal, interval, ratio, ...);
  - $v_k$ is the set of desired values of the parameter, or the constraint $<, \leq, =, \geq, >$ on the value of the parameter;
  - $u_k$ is the unit of the property value.

- $res_{i}^{Cost}$ specifies the contractual cost of use with respect to a particular cost ontology. Whatever the specific cost ontology, expected costs are likely to be specified as (at least) $res_{i}^{Cost} = \langle (n_1, t_1, b_1), \ldots, (n_r, t_r, b_r) \rangle$, where:
  - $n_k$ is the name of the CostParameter;
  - $t_k$ gives the type of the parameter (e.g., nominal, ordinal, interval, ratio, ...);
  - $b_k$ is the billing model ($pay per use$, $subscription$, ...).

- $res_{i}^{BeginTime}$ is the time the reservation of the resource starts for the particular contract;

- $res_{i}^{EndTime}$ is the time the reservation of the resource ends for the particular contract.
3.3 Resource Composition and Hierarchy

The ontology assumes a resources hierarchy in the sense that a Resource Object realizing a functionality or a Software Agent (which is a specialization of Resource Agent) realizing a capability can either be an assembly – i.e., a resource made of other resources advertising competencies – or be atomic – i.e., not made of other resources advertising competencies. This is materialized into the ontology by the composition association on the Resource Object and Software Agent concepts themselves (see Section 3). Similarly, Human Agents cannot be compositions of themselves but a HumanAgentsTeam is a composition of several HumanAgents; both of these concepts are specializations of Resource Agents so that a composition link joins these two concepts into the meta-model.

4 Case Study: Coking Process

The ontology is applied in this section within the realization of a particular service in the context of a supporting information system for a coking plant.

4.1 Context

CARSID, a steel production company located in the Walloon region, is developing a production management software system for its coking plant. The aim is to provide users, engineers and workers with tools for information management, process automation, resource and production planning, decision making, etc. Coking is the process of heating coal into ovens to transform it into coke and remove volatile matter from it. Metallurgical Coke is used as a fuel and reducing agent, principally by the blast furnaces, in the production of iron, steel, ferro-alloys, elemental phosphorus, calcium carbide and numerous other production processes. It is also used to produce carbon electrodes and to agglomerate sinter and iron ore pellets. The production of coke is only one step in the steel making process but details about the other phases of the production process are not necessary to understand the case study. For more information about this CARSID project, we refer the reader to [14].

4.2 Applying the Ontology

The illustration depicted in this section is in the context of the development of a software application made of several services; Pushing is the Service
Process realization we will be concerned with here. Indeed, other services are not essential for the present discussion and the reader should just keep in mind that the description that will be given here is part of a much larger set of developments.

In a few word, the Pushing service represents the process by which the Pusher Machine pushes the red-hot Coke out of the Oven through the Coke Guide into the Coke Car.

Typically, during a coke pushing process, the pusher machine situated at the front side of the ovens battery, removes the front door, pushes the red-hot coke into the coke car situated underneath the oven; the coke guide is used to remove the oven back door and to correctly guide the red-hot coke coming out of the oven into the coke car. The upper part of the machine runs a mechanic arm pushing slowly the coke out of the oven. The coke car is positioned behind the open oven receiving the red-hot coke pushed out of the oven by the pusher machine.

A formal set of functionalities and competencies is required to define this industrial process as reported in the Fulfillment statement of the following specification:

**Service Realization** Pushing

**Attribute** input: material in oven

  gc: GuideCoke
  cc: CokeCar

**Fulfillment**

\[
\text{cooking(input, o)} \land \text{align(pm, gc, cc, position(o))} \land \\
\text{guide(input)} \land \text{collect(input)} \rightarrow \diamond \text{cokeMaterial(cm)} \land \\
\text{emptyOven(o)}
\]

More precisely, \text{cooking(input, o)} represents the capability of the oven \text{o} to cook coke at a temperature between 1200 and 1350 °C during 16 to 20 hours to transform it into red-hot coke. Note that when this red-hot coke will be cold-down (through a passage at the quenching tower) it will be transformed into (metallurgical) coke, the finished product of the coking plant. Aligning the pushing machine, the guide coke and the coke car with the oven is a necessity to proceed with pushing the coke out of the oven, the reason why the functionality \text{align(pm, gc, cc, position(o))} is then specified. When proceeding with the pushing, the red-hot coke is guided by the coke guide (\text{guide(input)} capability) and collected into the coke car (\text{collect(input)} capability).

The list of resources used for achieving the required functionalities and competencies is given hereafter.
**Resource** Oven o

**Type:** Resource Agent

**Has**
- ovenId: O123, currentTemperature: 1250, maxCapacity: 10,
  - openDoorStatus: closed, exitDoorStatus: closed,
  - cookingStatus: active, cookingBeginTime: 18:00;

**Provides** evaluateCooking(input, this);

**Resource** PushingMachine pm

**Type:** Resource Agent

**Has**
- pmId: 1, maxPower: 10, status: available;

**Provides** align(this, position), pushing(input);

**Resource** cokeMaterial cm

**Type:** Resource Object

**Has**
- cokeId: lot1333, status: cooking;

**Provides** redHotCoke(input);

**Resource** guideCoke gc

**Type:** Resource Agent

**Has**
- gcId: gc1, status: available;

**Provides** align(this, position), guide(input);

**Resource** cokeCar cc

**Type:** Resource Agent

**Has**
- ccId: cc1, status: available;

**Provides** align(this, position), collect(input);

At runtime, the realization of the Pushing service implies the fulfillment of the **evaluateCooking**(input, this), **align**(pm, position(o)), **align**(gc, position(o)), **align**(cc, position(o)) and **collect**(input) capabilities leading to the series of contracts depicted hereafter.

**Contract** EvaluateCooking123

**Fulfills** evaluateCooking(input, o);

**Using** oven o, cokeMaterial cm;

**QualityLevel** N/A;

**Cost** N/A;

**Contract** Align123

**Fulfills** align(pm, position(o)), align(gc, position(o)),
  - align(cc, position(o));
Using pushingMachine pm, guideCoke gc, cokeCar cc, oven o;
Has beginTime: Date, endtime: Date;
QualityLevel N/A;
Cost Energy(pm(pos(x),pos(o)), gc(pos(x),pos(o)),
cc(pos(x),pos(o)));

Contract PushCokeLot1333
Fulfills collect(input);
Using oven o, cokeMaterial cm;
Has beginTime: Date, endtime: Date;
QualityLevel N/A;
Cost Energy(push);

4.3 Lessons Learned and Ontology Contributions

The use of the resource ontology as a pattern as well as a structured approach for resource representation and tracing among the analysis and design steps within the CARSID case study has allowed us:

- to improve the structural complexity. Indeed, the current application design has been enhanced with respect to metrics as the ones defined in [2];
- to provide a unified catalogue of available resources in the form of a yellow page system;
- to redefine resource as runtime dynamic elements. Object-oriented development used a static and passive description of the resources;
- to better understand resource utilization through the ex-post study of the contracts by production engineers. This allows to identify possible bottlenecks and re-optimize the production planning for adequate resource utilization;
- to compute and store into the caching system pre-positions for devices as the pusher machine, coke car and guide coke. Since several combinations of instances of these machines are available, the process is non-trivial;
- to express the process from a resource-based point of view which allows focusing on interoperability and evolutionary aspects.
5 Related Work

Different papers have proposed models to represent and monitor resources. Even heterogeneous resources have been dealt with – for example in [16] – their evocation has always been in the context of hardware resources. Indeed, [10] goes beyond the stage of defining an ontology for heterogeneous resources representation by proposing a complete architecture as well as an underlying management dimension with the use of a semantic repository. The proposal nevertheless only focuses on the particular context of grid computing and consequently hardware resources so that it cannot be of any help in terms of general business modeling. We nevertheless could envisage to include “our” ontology into their contribution to develop a larger resource monitoring system. Similarly [12] goes further in the idea of developing semantic resource management and within their management using a scheduling method but also remains in the context of hardware for grid computing. As evoked in Section 2, competency-based modeling for resource monitoring has been developed in [9] but in the specific context of human resources. Our ontological frame extends the competency concept to semantically encompass these two resource types. Conceptual models as i* [15] include the resource concept in their definition but do not define advanced frames for forward engineering such concepts at design stages; the ontological frame proposed here is intended to be extended fill this gap and explicitly define traceability between the business analysis and software design stages.

6 Conclusion

The new hardware resource sharing paradigm and distant software execution devices such as netbooks or smartphones paves the way to new heterogeneous resource allocation requirements supported by active software. To address this problem at best, an ontology that can easily and flexibly be included in a classical software engineering process, such as [14, 13], to develop resource-aware software systems has been presented in this paper. Section 2 has overviewed the structural choices of the ontology including the concepts of functionality and competency; the distinction between resources and actors and, finally, resource composition and hierarchy. Section 3 has presented the conceptual model itself while Section 4 has been devoted to a case study. Finally, Section 5 has briefly depicted related work.

The conceptual model in this research proposes to centralize resources’ offer and demand through the concepts of functionality and competency.
The ontology furnishes a common semantic for consumers (i.e. services) to rent resources that can themselves advertise their offer so that consumption contracts can be set up.

Future work includes extending the ontology with a process covering the whole software development life cycle from (agent-based) analysis to service level agreements. The development of a dynamic dimension allowing to document and implement a multi-agent system resources management as well as the realization on a case study in the field of outbound logistics is currently under progress.

References


