Proceedings of the International Workshop on Vocabularies, Ontologies and Rules for The Enterprise (VORTE 2005)

Article · January 2005

CITATIONS
0

READS
58

2 authors, including:

Giancarlo Guizzardi
Universidade Federal do Espírito Santo
234 PUBLICATIONS **4,096** CITATIONS

SEE PROFILE

Preface

These proceedings contain the accepted papers of the **International Workshop on Vocabularies, Ontologies and Rules for The Enterprise (VORTE)**, held in September 20th, 2005 in Enschede, The Netherlands.

This workshop, organized in the context of the **9th IEEE Enterprise Distributed Object Computing (EDOC) - The Enterprise Computing Conference**, can be considered as a second edition of the MDSW (Model-Driven Semantic Web) Workshop held at the EDOC 2004 in California, USA. However, besides the specific topics of Model-Driven Architecture (MDA) and the Semantic Web, the objectives of VORTE are: (i) to promote the discussion of the role of ontological research for enterprise engineering; (ii) bring together researchers from areas such as philosophical ontology, business process modelling, information modelling, software and domain engineering, and rule-based systems to support the interdisciplinary debate that this topic requires.

In accordance with the objectives of the workshop, we have assembled an international and highly qualified program committee, but also one on which different communities important for this enterprise are represented. As a result of the call for papers, this committee received 19 submissions from 16 countries worldwide and after a rigorous refereeing process, 9 high quality papers have eventually been chosen for presentation at the workshop and for appearing in these proceedings.

The material included in this volume reflects the nature of the forum that we want to promote with this edition of VORTE. The proceedings are structured in three concern areas, which are discussed in the sequel:

Part I (Foundational Ontologies): In the first part, we have three contributions that approach the topic of Enterprise Engineering from a foundational Perspective, i.e., by employing and advancing the theoretical work of areas such as Philosophy, Cognitive Sciences, Linguistics and Social Sciences. In the first of these contributions, Hannes Michalek addresses the topic of conceptual analysis of *causal relations* within the framework of the *General Formal Ontology (GFO)* developed by the OntoMed Research Group at the University of Leipzig. Following this article, we have the contribution by Emanuele Bottazzi and Roberta Ferrario discussing the foundations of an *Ontology of Organizations*. Bottazzi and Ferrario base their analysis in another foundational ontology named *DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering)* developed at the Laboratory for Applied Ontology (LOA) at the Institute for Cognitive Science and Technology, Trento, Italy. Closing this part, we have the *Context-Based Enterprise Ontology* proposed by Mauri Leppänen, aimed at promoting a common understanding of the nature, purposes and meaning of enterprise related concepts.

Part II (Enterprise Modeling): In this second part of the proceedings we have four contributions. In the first of these, Ricardo Falbo and Gleidson Bertollo present a formally defined *Software Process Ontology* that can be used by *Software Organizations* for understanding, communicating and reasoning about the Software Process Quality Domain. By showing how to interpret existing *Software Process Quality Standards* in terms of the proposed ontology, the article also supports software organizations in suitably employing these standards in their software process improvement efforts.

In the sequence, Slimane Hammoudi, Jérôme Janvier and Denivaldo Lopes present an approach inspired in the Ontologies and Database Literature for *explicitly differentiating between the concepts of mapping and transformation in MDA*. This approach can be used, for example, to derive *mapping specifications* between (platform independent) enterprise models and (platform specific) implementation models, which are independent of particular transformation definition metamodels (languages).

Still on this section, Kilian Kiko and Colin Atkinson elaborate on the need for *integrating existing enterprise information representation languages* aimed for human usability (in particular, *UML)*, with those aimed at machine processability and automatic reasoning (most notably, the description logics based *ontology representation languages* such as OWL). The authors discuss the strategies adopted by existing proposals in the literature and present their own view on the subject.

In an article concluding this part of the proceedings, Tae-Young Kim, Cheol-Han Kim, Jeong-Soo Lee, and Kwangsoo Kim describe an *MDA-based Enterprise Architecture Framework* that contributes to the configuration of process-centric and loosely-coupled *Virtual Enterprises*. The proposed framework harmonizes different existing approaches in the literature such as Enterprise Architecture (EA), framework-based development, meta-modelling and MDA to underpin the representation of enterprise models from different viewpoints, at different levels of granularity, generality and abstraction.

Part III (Business Rules): Finally, in this third part, we have two articles that target the topic of Business Rules. In the first of these articles, Rubby Casallas, Catalina Acero and Nicolás López discuss the *derivation of implementations from high level business rules* through successive *model transformations*. This work presents a modelling profile which defines a *vocabulary to model the concepts needed to integrate business activities and applications*, and, specifically, to assist the transformation of business process models to platform specific implementations. Concluding these proceedings, we have the contribution of Stijn Godertier and Jan Vanthienen, which proposes a rule-based approach to represent the semantics of data and control-flow perspectives in enterprise modeling. In particular, the authors show how a business rule model can be used to define and constrain the data elements of a *business vocabulary model* and the state transitions of a *business process model*. To this end, Goedertier and Vanthienen develop a *lightweight rule-based process ontology* and a *rule set ontology* for *rule-based business process modeling*, and demonstrate how a corpus of different kinds of Horn clauses can be generated from these models for *rule-based business process execution*.

There are several people that deserve our appreciation and gratitude for helping in the realization of this workshop. First, we would like to express our gratitude to both the program committee members and the additional external referees for their timely expertise in reviewing the papers, and to the authors for submitting their papers to VORTE. We would like to specially thank Colin Atkinson for accepting the invitation to deliver the keynote talk at the workshop. Moreover, we thank Roberta Ferrario, Emanuele Bottazzi and Claudio Masolo for their kind help in editing the proceedings. Finally, we are grateful to Elisa Kendall (chair of the MDSW 2004), Marten van Sinderen (general chair of EDOC 2005) and Bryan Wood (Workshop Chair of EDOC 2005) for their support in organizing this workshop as an EDOC Event.

September 1st, 2005

Giancarlo Guizzardi
University of Twente, The Netherlands and
Laboratory for Applied Ontology (LOA),
ISTC-CNR, Italy

Gerd Wagner
Institute of Informatics
Brandenburg University of
Technology at Cottbus, Germany

Program Committee

Aldo Gangemi	ISTC-CNR (Italy)
Andre Valente	Knowledge Systems Ventures (USA)
Andreas Opdahl	University of Bergen (Norway)
Andrey Naumenko	Triune Continuum Enterprise (Switzerland)
Brian Henderson Sellers	University of Technology Sydney (Australia)
Chris Welty	IBM Watson Research Center (USA)
Colin Atkinson	University of Mannheim (Germany)
Elisa Kendall	Sandpiper Software (USA)
Gerd Wagner	Brandenburg University of Technology at Cottbus (Germany)
Giancarlo Guizzardi	University of Twente (The Netherlands) & ISTC-CNR (Italy)
Joerg Evermann	Victoria University Wellington (New Zealand)
Michael Rosemann	Queensland University of Technology (Australia)
Michele Missikoff	IASI-CNR (Italy)
Mustafa Jarrar	Vrije Universiteit Brussel (Belgium)
Nicola Guarino	ISTC-CNR (Italy)
Oscar Pastor	Polytechnic University of Valencia (Spain)
Pericles Loucopoulos	University of Manchester (UK)
Ricardo Falbo	Federal University of Espirito Santo (Brazil)
Roel Wieringa	University of Twente (The Netherlands)
Terry Halpin	Northface University (USA)
Uwe Assmann	TU Dresden (Germany)
Robert Colomb	University of Queensland (Australia)
York Sure	University of Karlsruhe (Germany)

Table of Contents

A Causal Relation Based on Regularity and Manipulability

Hannes Michalek
Institute for Informatics (IfI)
Institute for Medical Informatics, Statistics and Epidemiology (IMISE)
Research Group Onto-Med
University of Leipzig
michalek@informatik.uni-leipzig.de

***Abstract*— Modeling causality is a necessary precondition for doing reasoning that entails prediction or planning, or helps at giving causal explanations. Unfortunately, most approaches in computer science (artificial intelligence) focus on the formal part of the problem underestimating the conceptual analysis. We meet this challenge within the framework of formal ontology and present an analysis that yields a notion of causality based on regularity and manipulation, and that is conceptually adequate in the aforementioned areas of reasoning (prediction etc.) as well as in the realm of scientific practice.**

I. Introduction

CAUSAL knowledge, i.e. knowledge of causal relations, is the key in a formal knowledge base, if a machine (computer, software agent) shall be of any help at either

> **Prediction** The mountaineer weighs 120kg including his equipment. His screw-gate carabiner has a breaking load of 16kN. Will it break if he falls down 3.5 meters?
>
> **Planning** The mountaineer (120kg) will fall 5 meters maximum (according to the route description). What carabiner has to be chosen so that it will not break in the case of an accident?
>
> **Causal Explanation** The mountaineer died in a climbing accident. What caused his death?

We see that the power of prediction[1] is fundamental for both planning and giving causal explanations. The interrelation is as follows: *Prediction* means giving the outcome of some initial situation or initial state of affairs. *Planning* consists of looking for an initial state of affairs whose desired outcome can be predicted (probably in several steps). Giving a *causal explanation* is looking for an earlier initial state of affairs that the actual outcome can be predicted of.

So if a machine (a knowledge base) is to be useful in either prediction, planning or search for causal explanation, it must inevitably contain some sort of representation of causal relationships.

The strategies to tackle this representation problem in the field of computer science (artificial intelligence) are numerous. (cf. [1], [2] or [3]) Yet they share the following problem: They usually take some notion of causality for granted (which is rarely made explicit) and develop an – undoubtedly excellent – formalism to describe it. However, in knowledge representation, a good (i.e. consistent and effective) formalism is only one requirement. The other is *conceptual adequacy*: Does the developed formalism model causality correctly?

This leads to conceptual analysis, which is the domain of analytical philosophy, and you will find an impressive amount of literature on causality in this area of research. Unfortunately, there is no consensus on causality which we could simply adopt. Still, you can find three major branches called *regularity analysis*, *counterfactual analysis* and *manipulation analysis*. (cf. [4]) Each of these theories focusses on a single feature of the causal relation, whereas we believe that *all* of them must be included in an adequate concept of causality:

Regularity is the most obvious characteristic of causality and widely used in science: Doing experiments means generating certain initial conditions and checking for a certain result afterwards. If some research group finds they have generated the very same conditions as their "rivals" but did not detect the same result, the theory at stake is regarded as falsified.

Counterfactual dependency has historically been understood as a variation of regularity[2] and is commonly expressed by "Had [cause] c not occurred, [effect] e would not have occurred either." [6] Today, following David Lewis [7], this theory is commonly seen as an independent approach based on possible world semantics.

The condition that the alleged effect should not take place without the alleged cause is useful to rule out certain kinds of accidental regularity, e.g. the relation between an arbitrary "cause" and an "effect" that simply always takes place, thus trivially fulfilling the regularity condition. It is also part of our everyday usage of causal expressions: "If I had not eaten this fish, I would not suffer from tummy-ache now."

We affirm this intuition but argue that it is already covered by regularity analysis (cf. section II-C).

Manipulability covers the intuition that the effect might be changed by manipulating the cause. This condition allows to filter out some other erroneous "causalities", as e.g. succeeding

[1] We will use prediction (and planning) in the sense of "predicting nature" or "physical prediction", as we are primarily concerned with physical causation. Predicting the outcomes of an abstract algorithm ("terminates after twenty iterations") or the like is not covered here.

[2] cf. Hume's famous "Or in other words" [5] connecting his definitions of regularity and counterfactual dependency.

effects of a common cause that would be regarded as a cause-effect pair by simple regularity analysis.

Secondly, it fulfills the pragmatic requirement of e.g. engineering: Causal knowledge that fulfills the manipulation condition enables us to make things happen the way we want them to, which might be the very reason why mankind engaged in scientific activities in the first place.

In contrast to the philosophical debate, we are not so much focussed on ontological simplicity but rather take Ockham's razor to be an advice to accept a broader base (i.e. regularity *and* manipulability) in order to yield a theory that has all the benefits of the ones above without suffering from their problems.[3] However, we had to realize that these theories have serious gaps that rendered our initial aim to provide a definition impossible (cf. II-D). Eventually, we had to restrict ourselves to collecting the necessary features of a causal relation.

Here's a summary of our approach and its results:

- We argue that it is presentials (which are not extended in time, as opposed to processes/events) that are the primary causal relata. (cf. section II-A)
- We formally describe the necessary conditions of an adaequate causal relation which capture regularity, counterfactual dependency and manipulation. (cf. sections II-C and II-D)
- This basic causal relation between presentials is extended
  - to allow for processes as causal relata (II-E), or
  - to avoid "anti-transitivity" (III-A).
- Examining the necessary conditions, we show that our notion of causality fulfills the aim of being a basis for understanding prediction and, hence, planning and causal explanation. The latter being understood in either the traditional Hempel/Oppenheim way or in the way of Woodward/Hitchcock's theory of explanatory generalization. (cf. section III-B)
- Finally (III-D), we show that our concept of causality matches the epistemic characteristics of modern science (i.e. how to find causal connections), with the design of clinical trials as an example.

II. The Details

A. Presentials as Relata

The repertory of causal relata in the discussion of causality is overwhelming, but if their ontological nature is not the focus of discussion, it is usually assumed that it is events that are causally connected (cf. [9]), the reason being that "[...] events have a strong causal flavor, due to their tight relationship with the notions of change and time, and this makes them intuitively appealing causal relata.", as Lehmann et al. put it (cf. [10]), and we agree with these authors that everyday language prefers processes/events as causal relata.

Yet we think that serious problems arise if we don't take this everyday usage as a merely pragmatically justified abbreviation but as following a full-fledged ontological picture, our starting point being the **problem of causal relevance**:

Think of the simple situation where one billiard ball moves in the direction of another ball which rests on the cloth. The two balls touch, and the second ball's movement begins (while the first ball changes speed and/or angle). In terms of events, this situation would be analyzed as two processes meeting at the time of contact. A problem arises if you ask which part of the first process is causally relevant to the second one - the first half or the second?[4]

Think of the first half alone. There is no collision, thus, nothing causes the second ball's movement. But if we take the second half alone (or combined with a different first half), the result would be just the same as in the unmodified situation. The second half seems to bear *all* the causal power. Disregarding the first half of the first process, we can pose the same question for the second half alone: Which quarter of the first process is causally relevant, the third or the fourth? Again, we find that only the fourth would lead to the same movement of the second ball. Moving on, we must say that it is the last eighth that bears all the causal relevance, the last sixteenth, and so on.

This leads us to the assumption that it is the situation or state of affairs at the very end of process one that is causally relevant to the second process.

A possible objection to this argument is: Splitting up causal relevance is not allowed; it is always the *whole* process that causes an effect. But think of scientists making experiments to check the predictions of a theory. They will do so by creating the initial conditions the theory is about. Notice that they do not have to create these conditions in a defined way. If a certain low temperature is needed, they are free in choosing the means of cooling. It seems sensible to regard the way of cooling as irrelevant to the effect.

The same holds for the billiard balls: It is of no relevance to the second ball's movement *how* the first one got his velocity or angle. It could have been struck by the queue, hit by another ball, thrown by somebody or moved to its place by a complex apparatus. In all of these cases, the second ball would behave in exactly the same manner as long as the *situation at the moment of touching* is the same.

An analogous thought experiment can be applied to the second process. It can turn out in very different ways: The second ball could run down the table and eventually come to a hold due to friction. It could also be stopped seconds by a obstacle or even accelerated if the table is not level, etc. This again leads us to the assumption that it is not the whole process that is caused but just the state of affairs at its first time-boundary.

According to the formal ontology of GFO[5], presentials are the kind of entity that exist wholly at time-boundaries (which

[3] Marilyn McCord Adams puts the *Anti-Razor* this way: "[...] where fewer entities do not suffice, posit more!" [8]

[4] This argument is based on an idea taken from Michael Jubien [11].

[5] General Formal Ontology, being a part of the ontological research of OntoMed at the University of Leipzig. This paper is a contribution to the work done there, as e.g. presented in [12].

is GFO's term for time points or time slices), so we take presentials as being the primary causal relata.[6]

The formal account is given at the end of section II-B.

B. Temporal Structure

Referring back to the billiard balls, we plead for a very special temporal relation between the causal relata which is neither simple succession nor simple synchronicity, but – as we will argue in the following – gives a good answer to the **question of temporal and causal connection**:

Taking the process-view again: How should the two processes be connected? If time is modeled as intervals of real numbers, there is a well-known problem. There are four possibilities which do not suffice:

- The first interval is closed on the right, the second one on the left. Where the two intervals either overlap, which is not the kind of "immediate succession", we wanted to model. Or where they do not overlap, which implies a gap between cause and effect.
- The first interval is closed on the right, the second one open on the left. This is mathematically sound, but what should a process with an open boundary mean? Contrary to the movement of the second ball, this process would have no defined beginning.
- The first interval is open on the right, the second one closed on the left. Again, the model differs to the real movement of the ball with respect to the defined end of the process.
- The first intervall is open on the right, the second one open on the left. This time, neither would the first process have a definite end nor the second one a defined beginning.

Fortunately, there are other approaches to time and GFO provides means to tackle this problem by modeling time not by intervals of real numbers but by chronoids with time-boundaries that can *coincide*, which means that they are "in the same place" while still being distinct. So we have a true "end point" of the first process: time-boundary t_c, and a true "starting point" of the second process: time-boundary t_e, with t_c and t_e coinciding, therefore without a gap. (cf. [12])

This matches our idea that it is the relation between presentials that process causality is based upon. Processes are causally connected, if the presentials that exist at the coinciding time-boundaries t_c and t_e are causally related. (More on process causality in II-E.)

Our discussion of the nature of the relata and the temporal structure can now be summarized as follows:

$$\begin{aligned} eq.time(x,y) =_{df} \quad & \exists p_1, p_2, t_1, t_2 \\ & (Proc(p_1) \wedge Proc(p_2) \wedge rtb(t_1, p_1) \wedge ltb(t_2, p_2) \\ & \wedge Pres(x) \wedge Pres(y) \wedge at(t_1, x) \wedge at(t_2, y) \\ & \wedge coinc(t_1, t_2)) \end{aligned}$$

$$cause(x,y) \rightarrow \quad eq.time(x,y)$$

[6]Another line of argument (which shall not be carried out at full length here) is, that is it objects, and not processes that have masses or speed etc. which seems to be necessary, to play the role of "puller and shover and twister and bender". (Expression taken from [13])

with $rtb(t_1, p_1)$ expressing that t_1 is the right time-boundary of the chronoid which is framing process p_1 and $ltb(t_2, p_2)$ expressing that t_2 is the left time-boundary of the chronoid which is framing process p_2.

C. Regularity

In order to speak of regularity at all, we first need a collection of *similar* incidents. Our idea is to capture this similarity by means of universals. So regularity's rough motto "similar scene followed by similar scene" is modeled by instances of universals.

Secondly, we need some rule on these instances. And as we want to cover chances as well as 100% regularity, we take statistical dependency to connect the existence of the instances.

$$\begin{aligned} Reg(x,y) =_{df} \quad & \exists U_1, U_2 \\ & (Univ(U_1) \wedge Univ(U_2), x :: U_1, y :: U_2 \\ & \wedge L(U_1, U_2)) \end{aligned}$$

$$cause(x,y) \rightarrow \quad Reg(x,y)$$

with $L(U_1, U_2)$ expressing the *statistical dependency* between the instances of U_1 and U_2: If we take all pairs (right time-boundary, left time-boundary) of – as $eq.time()$ requests – coinciding time-boundaries and the presentials existing at these boundaries, the probability of finding an instance of U_2 at the left time-boundary in such a pair is lower than the probability of finding an instance of U_2 at a left time-boundary in a pair where there is an instance of U_1 at the coinciding right time-boundary.

We believe that this also captures the counterfactual intuition: If L is a 100% dependency, the effect does not happen without the cause, and if L expresses that the probability of the effect is higher with the cause happening, this implies that it is lower without. In other words: The effect would not have happened or would have been less likely to happen, which is how we introduced the the counterfactual intuition in the introduction.

More precisely: Nothing *like* the actual effect would have happened. This is an important difference to modern Lewis style counterfactual analyses as presented e.g. by John Collins [6]. Those theories treat the intuition of "the effect would not have happened" as being about another possible world where *the very same* effect – the one that did happen in the actual world – does not happen.

D. Manipulation

At this point, there is one last requirement left: The manipulation condition. The effect must be manipulable by the cause. Let us come back to the billiard balls example: In what way is the second ball's movement manipulable by the first ball's movement?

Firstly, note that the kind of difference in cause as well as effect must fulfil certain conditions to be reasonable. As James Woodward puts it: "...we have no coherent idea of what it is to change a raven into a lizard or kitten". [14] So we lack any causal intuition about a situation where the first ball is

changed, say, into a copper wire. And the same holds true for a "change"[7] from mass to color. We believe that it is changes in *quality values*[8] that manipulation is about. The second ball's velocity or angle of movement can be changed by changing the first ball's movement. This means that the causality relation's necessary conditions must include qualities and quality values of the relata:

$$p.has.v(x, Q_x, v_x) =_{df} \quad \exists q(Qual(q) \wedge q :: Q_x \wedge has\_quality(x, q) \wedge has\_value(q, v_x)$$

$$Man(x, Q_x, y, Q_y) =_{df} \quad \exists x', y' ((comp(x, x') \wedge comp(y, y') \wedge p.has.v(x, Q_x, v_x) \wedge p.has.v(x', Q_x, v'_x) \wedge p.has.v(y, Q_y, v_y) \wedge p.has.v(y', Q_y, v'_y) \wedge (v_x \neq v'_x)) \rightarrow (v_y \neq v'_y))$$

$$cause(x, y) \rightarrow \quad \exists Q_x, Q_y(Man(x, Q_x, y, Q_y))$$

A few words on the definitions: $p.has.v(x, Q_x, v_x)$ connects the quality value v_x to the presential x, with v_x belonging to a certain quality universal Q_x. $Man(x, Q_x, y, Q_y)$ demands for at least one pair of non-equal values (of the same universal) on the side of the cause which are reflected in a pair of non-equal values on the side of the effect.

The crucial relation, however, is comparability $comp(x, y)$, whose intension is easy to understand: Not every pair of presentials on the side of the cause that differ in a quality value qualifies to take place in the manipulation condition. Take the movement of the first ball. We have already seen that exchanging the presential at the last time-boundary of that process by something that is not a ball at all does intuitively not count as a relevant manipulation. Furthermore, the replacement by a presential that is not "coherently" connected to the process of the ball's movement seem to be problematic, too.

Unfortunately, this problem is not addressed in causal literature on manipulability. Its origin can be found in the history of the manipulation condition. Early theories used to refer to "changes due to human interaction", which would have been a rather strong constraint to the allowed manipulation, but this apparently failed in causal situations where human powers were too limited to have any influence as in the relation between moon and tide or between earthquakes and plate tectonics. So newer theories try to avoid this problem by defining the allowed manipulations without reference to human interaction, which now seems to be too thin a concept to discriminate between relevant manipulations and irrelevant ones. (cf. [14] for history of the manipulation theory.)

Woodward/Hitchcock present a notion of manipulability that introduces an exogeneous "intervention" that changes the cause without influencing the effect directly (i.e. via a route that excludes the cause).[9](cf. [15]) This implies that the presential at the end of the first process is always bound to a process, but the problem is just transferred to the question of "What processes are allowed?", which in turn should exclude the exchange of single presentials that render the process "unnatural", "strange" or "incoherent", as it could be referred to.

Again, we stress that the notion of an allowed manipulation is by intuition sufficiently clear to be used here. Going back to the billiard balls, a reasonable manipulation could be slowing the first ball down by hand. This manipulation would set the value of the first ball's velocity "in the right way", in terms of Woodward/Hitchcock's approach: without influencing the second ball via a route that excludes the first ball.

Slowing down the first ball by a big fan that creates strong wind on the billiard table would surely not fulfill the manipulation condition as it directly influences the second ball as well.

Summarizing, we understand the basic causal relation to have the following three necessary features:[10]

$$cause(x, y) \rightarrow \quad eq.time(x, y) \wedge Reg(x, y) \wedge \exists Q_x, Q_y(Man(x, Q_x, y, Q_y))$$

E. Processes as Relata

We started our analysis by showing that it is reasonable to take presentials as primary causal relata, yet we agree that it is quite common (and useful!) to attribute causal relations to processes. We will now show, how our presential based causality relation can easily be extended to cover processes as causal relata as well.

All that is required is the aforementioned relationship between presentials and processes: Processes have boundaries, and presentials are the kind of entity that exists wholly at such a single time-boundary. Following our analysis of the billiard balls example in II-A, we introduce process-process causality by means of the presentials existing at the processes' boundaries:

$$at.end(x, y) =_{df} \quad Pres(x) \wedge Proc(y) \wedge \exists t(rtb(t, y) \wedge at(t, x))$$

$$at.beginning(x, y) =_{df} \quad Pres(x) \wedge Proc(y) \wedge \exists t(ltb(t, y) \wedge at(t, x))$$

$$p.cause(x, y) \rightarrow \quad \exists u, v (Proc(x) \wedge Proc(y) \wedge Pres(u) \wedge Pres(v) \wedge at.end(u, x) \wedge at.beginning(v, y) \wedge cause(u, v))$$

[7]Note that "change" is not understood as a difference in the very same presential but as a difference between "comparable" presentials. This has two consequences: First, the question of "comparability" is raised, which we shall discuss in the following. Secondly, our approach also handles "static causation", which is problematic for approaches that are based on changes (e.g. cf. [10]).

[8]Quality values in GFO are, in short, the values of a property's instances: Weight is the property, this volleyball's weight is the particular quality of the particular volleyball and 260g is this quality's value.

[9]This intervention condition on the manipulation solves a problem which Lehmann et al. merely "defined away": There are qualities like shape and what they call "location", that always change together ("structural constraints", cf. [10]. With the intervention condition at hand, we see that, in this case, there is no manipulation that changes shape without directly influencing the location. Thus, there is no causal relation. The connection might probably be analyzed as "conceptual overlapping".

[10]If it were not for the problems with the manipulation condition (cf. II-D), you could read $=_{df}$ instead of $\rightarrow$ here.

Causality between processes is, thus, defined by causality between the presentials at the "meeting point" and can be used to express immediate cause. For expressions that include chains of causally connected processes we will need a further modification of $p.cause(x, y)$, cf. III-A.

Analogously, we can define the rather technical relations $p_1.cause(x, y)$ between a process and a presential and $p_2.cause(x, y)$ between a presential and a process, which are useful for expressing that a process causes a certain state or that a certain state of affairs caused a process:

$$\begin{aligned} p_1.cause(x,y) =_{df} \quad & Proc(x) \wedge Pres(y) \\ & \wedge \exists u(Pres(u) \wedge at.end(u,x) \\ & \wedge cause(u,y)) \end{aligned}$$

$$\begin{aligned} p_2.cause(x,y) =_{df} \quad & Pres(x) \wedge Proc(y) \\ & \wedge \exists v(Pres(v) \wedge at.beginning(v,y) \\ & \wedge cause(x,v)) \end{aligned}$$

Referring to these relations as "technical" does not mean that they have no ontologically sound interpretation. They are just not that common in ordinary language and must not be confused with expressions that merely state the situation at the end or at the beginning of a process ("Running the whole way made her be at the office at 10:00pm.") without this end or beginning being causally connected to another presential.

III. Consequences and Discussion

The last section illustrated how we understand our theory to fulfil the regularity, counterfactual and manipulation condition. We will now discuss various consequences of our approach.

A. Reflexivity, Symmetry, Transitivity

Where does $cause(x, y)$ stand in terms of reflexivity, symmetry and transitivity? We will give some remarks here, with transitivity bearing special importance.

Reflexivity: As $eq.time(x, y)$ is defined by two presentials existing at distinct time-boundaries, $cause(x, y)$ is irreflexive, but note that the technical term "reflexivity" refers directly to the presentials at stake.

If you take the GFO-route to the problem of identity through time, there are special universals ("persistants") whose instances are those presentials that we refer to as "being the *same* thing/object/person through time". Identity through time is provided by the universal, whereas the concrete presentials are not identical. (cf. [12])

Thus, what $cause(x, y)$ allows for is that x and y of $cause(x, y)$ are instances of such a persistant. If that is the case, an *object* can have a causal connection to "itself", meaning that one instance of this object's persistant is causally related to another instance of the same persistant. This means that we can have reflexivity on the level of objects without having reflexivity on the level of presentials.

The following example may illustrate that the notion of an object being causally related to itself is useful: Consider an object moving in vacuum without being influenced by any kind of force. Our approach allows for cutting the movement into two parts, calling the first the cause of the second one, which seems to be a reasonable answer to the question of what caused the second part of the movement.

In terms of physics, the reason for the object's constant movement is captured by the preservation of impulse (together with preservation of energy). This law of physics fulfills our conditions of regularity and manipulability.

Symmetry: Once more, $eq.time(x, y)$ needs exactly one left and one right time-boundary, so $cause(x, y)$ is asymmetric, which is desirable to prevent the effect causing the cause, as this would ruin the notions of cause and effect entirely. Secondly, symmetry would have included so-called "backward causation", which is commonly taken to be counterintuitive, at least in the case of causal loops (or time-travel).[11]

Transitivity: Again, it is the time-boundaries which now lead to what could be called "anti-transitivity": $cause(x, y) \wedge cause(y, z) \rightarrow \neg cause(x, z)$. Still, we acknowledge that causal transitivity is commonly assumed in everyday life, so rather than ruling it out we want to make explicit what conditions must be fulfilled for a transitive usage of causality. Introducing a new relation $t.cause(x, y)$ analogous to $cause(x, y)$ based on temporal succession ($x <_t y$) instead of coincidence and allowing for left time-boundaries as well as right time-boundaries should provide a good start:

$$\begin{aligned} succ.time(x,y) =_{df} \quad & \exists p_1, p_2, t_1, t_2 \\ & (Proc(p_1) \wedge Proc(p_2) \\ & \wedge((rtb(t_1,p_1) \wedge ltb(t_2,p_2)) \\ & \quad \vee(rtb(t_1,p_1) \wedge (rtb(t_2,p2))) \\ & \quad \vee(ltb(t_1,p_1) \wedge (rtb(t_2,p_2))) \\ & \quad \vee(ltb(t_1,p_2) \wedge (ltb(t_2,p_2)))) \\ & \wedge Pres(x) Pres(y) \wedge at(t_1,x) \wedge at(t_2,y) \\ & \wedge(t_1 <_t t_2)) \end{aligned}$$

$$\begin{aligned} t.cause(x,y) \rightarrow \quad & succ.time(x,y) \\ & \wedge Reg(x,y) \\ & \wedge \exists Q_x, Q_y(Man(x,Q_x,y,Q_y)) \end{aligned}$$

Again, we could extend this relation to allow for processes as relata, the new relations looking like their counterparts from II-E with "$cause()$" being replaced by "$t.cause()$".

Note that $t.cause(x, y)$ still fulfills the regularity and manipulation condition.[12] With $t.cause(x, y)$ in the background, $cause(x, y)$ can be seen as the most immediate kind of a $t.cause(x, y)$ connection, but we hesitate to take it as the more basic one as it allows for (big) temporal differences between cause and effect.

Let us come back to the question of transitivity: We introduced $t.cause(x, y)$ to get rid of the "anti-transitivity" of $cause(x, y)$. We succeeded in that, but it is important to see that we still have no transitivity, which is due to the manipulation condition. Regularity is transitive, but "having a manipulation for" $t.cause(a, b)$ as well as $t.cause(b, c)$ does not entail having a manipulation for $t.cause(a, c)$.

[11] According to J. Faye, backward causation should not be confused with causal loops; none of them entails the other. (cf. [16])

[12] With the regularity – still expressing statistical dependency – slightly adjusted in order to not depend on coinciding time-boundaries any more.

Yet, in certain everyday situations, a manipulation "spanning" over a and c could be obvious: "My kicking the ball caused the window to shatter." is a reasonable sentence, which presupposes transitivity as there are more that two processes involved. Let us assume there are three of them: the kicking (k), the movement (m) of the ball, and the shattering (s) of the window. Between both pairs holds the process-causality relation: $p.cause(k, m)$, $p.cause(m, s)$. And we can easily think of a manipulation of k that yields a different s without influencing the effect directly. We could e.g. change the speed or the direction of the kicking. This manipulation fulfills the conditions for the transitive version of process-causality $t.p.cause(k, s)$, which would justify the everyday inference.

It might well be, that certain relations between the manipulations can provide transitivity, e.g. when manipulation m_1 on a $p.cause()$ yields an effect that is itself a manipulation m_2 to another $p.cause()$[13], which is to be elaborated in the future.

B. Prediction, Planning, Causal Explanation

How is (causal) prediction to be understood in out framework? As introduced at the beginning of this paper, it is about telling a future state (F) on the basis of an earlier one (E). It is easy to see that our causal relation allows for doing that: If the knowledge base includes e.g. $t.cause(E, F)$ and E, regularity allows for inferring F's existence while manipulability allows for inferring some of its quality values.

Planning (i.e. the question of what to do in order to yield a certain result) is based on prediction, and our notion of causality actually gives two interrelations: First, as given in the introduction, if we have a range of initial sequences from which we can predict their outcomes, we can choose the one with the outcome we like best. This is a consequence of regularity. But secondly, causal knowledge contains knowledge about manipulation, which also helps achieving the desired result.

Giving a causal explanation again uses the connection between earlier causes and later effects, but this time, the effect has already taken place and we look for an earlier situation that is of a kind which results in the effect as it actually happened. In our terms, you search for earlier processes/presentials that have a causal connection to the actual effect.

Concerning explanation, there is another interesting consequence of our approach:

C. Explanatory Power

The classical theory of explanatory power of generalizations is given by Hempel/Oppenheim and their deductive nomologic approach that – in short – connects the explanatory power of generalizations to their expressing a law. (cf. [17]) If one takes this theory of explanation for granted, our regularity requirement already accounts for "telling the cause" being useful as an explanation. However, there is discussion about the deductive nomologic approach and recently, Woodward/Hitchcock [18] presented another theory on the explanatory power of generalizations.

The problem is that there are generalizations that fulfill a notion of lawfulness, like "All people in this room have black hair". But with regard to the question "Why does this person have black hair?", the answer "Because he/she is in this room" is not very convincing.

Woodward/Hitchcock's theory proposes the following: A generalization has explanatory power if it provides answers to a range of "what-if-things-had-been-different questions". [18]

With regularity and especially manipulation at the core of causality, we see that our notion of causality is well-suited to answer these questions.

D. Scientific Research Practice

Natural science is *the* human enterprise to find causal relations. An adequate theory of causality should match scientific research practice, which obviously *is* successful in discovering causal connections.

We have already seen the connection between falsification and the regularity requirement of our causation approach (cf. section I). While physics, as an example, makes frequent use of strict falsification, there are other branches of science that (due to the complexity of the field) work in a different way. For instance, testing hypotheses on the effects of drugs requires another kind of experiment using *test and control groups*. This roughly means that there are two groups of patients that share a disease, only one of them being treated with a certain drug.

How can this procedure be explained by our theory of causality?

- Firstly, there is simple testing for regularity: Is recovery statistically dependent on giving the drug?
- Secondly, if there is statistical dependency, the design of the study has to make sure *what* made the difference between the two groups. Then the manipulation condition is at stake: What accounts for making the differences in the cause must not be connected to the effect (except by the route via the cause). The best example for not fulfilling this condition is the placebo effect that connects recovery directly to treatment without the drug being involved.
  Medical research has developed a vast variety of procedures (blinded, double blinded etc.) to avoid such side-effects of the treatment, which fit our manipulation condition perfectly.

E. Open Questions, Further Development

We believe that there are several fields in which our approach should be examined in more detail and we will collect those loose ends here:

- First of all, there are the different extensions of the causal relation that were introduced in this paper. We think that their characteristics should be investigated more thoroughly in order to give simple and systematic instructions to the knowledge engineer on when to use which.

[13] cf. the Woodward/Hitchcock "intervention" in II-D that allows for the manipulation being causally related to the first relatum.

- A finer analysis of the manipulation condition is badly needed. This would contribute to the philosophical discussion as well.
- We believe our approach to cover physical causation, but we are quite positive that the main ideas – regularity and manipulation – are common to causation between other entities as well (negative causation, which we did not mention here, put aside), be it even in the disputed field of psycho-physical causation. Obviously, the necessary conditions would have to be adjusted to the kind of entity that plays the role of presentials in the mental stratum, but regularity and manipulability seem to be of crucial importance to call something a cause at all.
- Our approach aims at describing the features the relation between cause and effect has to fulfil in order to justly be called a "causal relation". We did not address the problem of which of the possibly numerous causes may be called *the* cause and we tend to think that this is a psychological question rather than an ontological one.
- What we did not address, either, is the question of how *multiple* causal relations to the same (effect-)presential will turn out. We think they might be "summed up" like forces in physics with the result following the strongest cause, being a mixture of several effects, or not being visible at all, in case the causes neutralize each other's effects.

ACKNOWLEDGEMENTS

Thanks for substantial help and stimulating discussion go to the Onto-Med research group at the University of Leipzig, especially to Heinrich Herre, Frank Loebe and Patryk Burek.

REFERENCES

[1] N. McCain and H. Turner, "Causal theories of action and change," in *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, 1997.

[2] A. Bochman, "A logic for causal reasoning," in *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03)*, G. Gottlob and T. Walsh, Eds., 2003, pp. 141–146.

[3] J. Pearl, *Causation*. New York: Cambridge University Press, 2000.

[4] J. Kim, "Causation," in *The Cambridge Dictionary of Philosophy*, 5th ed., R. Audi, Ed. Cambridge, United Kingdom: Cambridge University Press, 1998, pp. 110–112.

[5] D. Hume, *An Enquiry concerning Human Understanding.*, 1748.

[6] J. Collins, E. Hall, and L. Paul, *Causation and Counterfactuals*. Cambridge, Mass: MIT Press, 2004.

[7] D. Lewis, "Causation," *Journal of Philosophy*, vol. 70, pp. 556–67, 1973.

[8] M. McCord Adams, "Occham's razor," in *The Cambridge Dictionary of Philosophy*, R. Audi, Ed. Cambridge University Press, 1995.

[9] J. Schaffer, "The metaphysics of causation," in *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed., Spring 2003. [Online]. Available: http://plato.stanford.edu/archives/spr2003/entries/causation-metaphysic%s/

[10] J. Lehmann, S. Borgo, A. Gangemi, and C. Masolo, "Causality and causation in dolce," in *Formal Ontology in Information Systems, Proceedings of the International Conference FOIS 2004*, 2004.

[11] M. Jubien, *Contemporary Metaphysics*. Oxford, Malden (Mass.): Blackwell Publishers, 1997.

[12] B. Heller and H. Herre, "General ontological language (GOL) – A formal framework for building and representing ontologies. Version 1.0," *Onto-Med Report (University of Leipzig)*, no. 7, 2004.

[13] J. Bennett, *Events and their Names*. Indianapolis: Hackett Publishers, 1988.

[14] J. Woodward, "Causation and manipulability," in *The Stanford Encyclopedia of Philosophy (Fall 2001 Edition)*, E. N. Zalta, Ed., 2001, http://plato.stanford.edu/archives/fall2001/entries/causation-mani/.

[15] J. Woodward and C. Hitchcock, "Explanatory generalizations, part I: A counterfactual account," *Nous*, vol. 37, no. 1, 2003.

[16] J. Faye, "Backward causation," in *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed., Winter 2001. [Online]. Available: http://plato.stanford.edu/archives/win2001/entries/causation-backwards/

[17] C. G. Hempel, *Aspects of Scientific Explanation and Other Essays in the Philosophy of Science*. New York: Free Press, 1965.

[18] C. Hitchcock and J. Woodward, "Explanatory generalizations, part II: Plumbing explanatory depth," *Nous*, vol. 37, no. 2, 2003.

A Path to an Ontology of Organizations

Emanuele Bottazzi* Roberta Ferrario*

***Abstract*— This paper presents a preliminary proposal of an ontology of organizations based on DOLCE (*Descriptive Ontology for Linguistic and Cognitive Engineering*). An ontological analysis of organizations is the first, fundamental and ineliminable pillar on which to build a precise and rigourous enterprise modelling. An ontological analysis makes explicit the social structure that underlies every organizational settings. In particular, the paper tries to explain what are organizations, roles and norms, how they are interrelated, what it means for a norm to be valid in an organization and what it means for an agent to be affiliated to an organization.**

I. INTRODUCTION

THE aim of this paper is to lay down the bases for an ontological analysis of organizations.

Obviously, there are many possible ontologies of organizations, based on different theories of organizations; therefore, our analysis is biased in two senses: it is influenced by the philosophical assumptions we take (relying on the literature and on our personal intuitions) and by the formal framework we used, which is itself based on other more general assumptions. Nevertheless, this should not be regarded as a drawback of the proposal, but rather as an ineliminable feature of all proposals of this kind.

Many kinds of analysis can be and have been conducted on organizations, so it is important to understand what an ontological analysis is and how it can be distinguished from other kinds of analysis.

A first distinction that can be traced is relative to the focus of the analysis that can be either on dynamic or on static aspects of organizations. Among analyses of the dynamics of organizations we can further distinguish what can be called "genetic analyses" from "analyses of the actions".

Generally speaking, genetic analyses have the purpose of answering to questions like: How are organizations born? What happens when an organization is born? What is necessary in order for an organization to be born? What kind of relation does it entartain with its founders? These questions, although very important, are not adressed by the ontological analysis we want to pursue in the paper.

On the other hand, important questions for an analysis of actions are: How are collective actions performed? Which relations do they entertain with actions of the individuals who participate to the collective one? Can organizations be considered agents of some kind? and, if this is the case, How can they act in the world? Are they responsible for their actions? What can or cannot they do? All these questions are in a way peripheral to the ontological analysis, but some of the answers can be indirectly inferred by the answers to the central ontological questions.

* Laboratory for Applied Ontology
Institute for Cognitive Sciences and Technologies
National Research Council
via Solteri 38
I-38100 Trento
Phone: +39 0461 436639, e-mail:{bottazzi,ferrario}@loa-cnr.it

These central questions mainly concern the so called static aspects of organizations. Such questions are: Which kind of relation does it hold between an organization and its members? What is necessary for a certain agent in order for him/her to be a member of an organization? Which is the relation holding between the roles in an organization and its normative layer? In other words, what is important for this analysis is to isolate the fundamental entities of the social/organizational domain and to characterize the relations holding among them, taking them – in some sense – for given, thus without considering their origin.[1] Along these lines, in this paper we will especially underline the importance of norms in determining the nature of social entities and relations in the internal dimension (among members inside the organization and between organizations and their members) rather than in the external one (among different organizations).

An ontological analysis of organizations is the first, fundamental and ineliminable pillar on which to build a precise and rigourous enterprise modelling. An ontological analysis makes explicit the social structure that underlies every organizational settings.

The study carried out in this paper will rely on DOLCE (*Descriptive Ontology for Linguistic and Cognitive Engineering*) [1], an already existing foundational ontology that has been developed at the Laboratory for Applied Ontology (LOA) of the Institute for Cognitive Sciences and Technology of the Italian Research National Council.

DOLCE has proven very useful in adressing various problems and the paper is part of a collection of works aimed at extending DOLCE as to make it suitable for many distinct specific domains.

II. BACKGROUND CONCEPTS

As already mentioned in the introduction, this work is part of a larger project aimed at extending the DOLCE ontology as to comprise also the social dimension. This effort has already been started with the papers [2] and [3] and we will try to reuse and integrate them in the present paper.

The notions of DOLCE we will use in the paper are those of endurant, perdurant, time location, agentive social object and non agentive social object. Endurants and perdurants are two of the most basic categories of DOLCE; *endurants* are entities that are in time, like me, my cat, an umbrella, a flower (so, roughly speaking, they correspond to the commonsensical notion of object), while *perdurants* happen in time (they can be assimilated to the commonsensical events) and examples of them are conferences, tennis matches, my sister's wedding etc.

With respect to *social objects* (both *agentive* and *non agentive*), we can intuitively say that they are objects (endurants) produced by communities, in the sense that they depend, for

[1] A further possible kind of analysis is the teleological one, namely the study of the relations that organizations have with their goals; this aspect is certainly relevant from an ontological standpoint, but it will not be adressed in the present work, due to the fact that it deserves a long and detailed inquiry, not possible in the limited lenght of a paper.

their existence, on intentional agents that conventionally create them and accept them. They can be divided in agentive or non agentive on the basis of their possession of intentionality. Examples of agentive social objects are legal person and customer, while examples of non agentive social objects are a law or a currency.

Starting from the notion of non agentive social object, [2] has given the definition of some more specific notions, like that of social concept, of description[2] and of social role.

Social concept and *description* are two disjoint subcategories of the category "non agentive social object" and they are connected by a *definition* relation. This should give the intuition that social concepts are contextual in nature and descriptions are the context in which they are defined. In addition to what already stated about non agentive social objects, we can say that descriptions are always encoded in at least one physical support; they begin to exist when they are firstly encoded and continue to exist until the last physical support in which they are encoded is destroyed and, finally, one and the same description can be expressed in many different ways and languages without losing its identity (provided its semantic content doesn't change).

Another relevant feature characterizing social concepts is the relation (called in [2] *classification*) that these entertain with categories of the so called "ground ontology", namely categories that are taken to be not contextual (in other words, not social). As an example, take the concept "crown of the king of Spain"; in this very moment there's probably a piece of precious metal that is classified by this concept, but this relation is given by the fact that there's a description (the one of the kingdom of Spain) defining the concept of "crown of the king of Spain". We can notice that this concept doesn't necessarily classify always the same object, in fact probably 200 years ago another piece of metal, possibly made up of a different precious metal, was classified by the very same concept. Moreover, it is possible that in a certain moment a concept ceases to classify at all, for example if Spain becomes a Republic, or like at the present moment the "crown of the (actual) king of Italy", which doesn't classify anything.

In some sense, apparently the objects of the ground ontology – that we pretend to be acontextual – and the social objects – whose contextual nature is explicitly taken into consideration – belong to two different and heterogeneous domains but, in line with [2], both for technical reasons[3] and for pragmatic reasons[4], we put ground objects, social individuals and social concepts as well at the same ontological level. So, intuitively, we can say that social concepts are like properties, and thus treated as first class citizens in our ontological framework.

Social roles are instead a subclass of social concepts, with two additional features, that in [2] have been called *anti-rigidity* and *foundation*. Anti-rigidity expresses the fact that roles have dynamic properties and it establishes that "for any time an entity is classified under it [a concept], there exists a time at which the entity is present but *not* classified under the concept" [2]. Foundation, on the other hand, is the property that shows the relational nature of roles; in fact, it states that "A concept x is founded if its definition involves (at least) another concept y (definitional dependence) such that for each entity classified by x, there is an entity classified by y which is external to it (generic existential dependence on external properties)" [2].

Other two notions we want to use as backbones for our proposal have been presented in [3], where a very rich axiomatization that we will not present here is given; these are the notions of collectives and collections.

Very generally, we can say that *collectives* are collections of intentional agents. *Collections*, on their turn, are social objects that generically depend on their members (in the sense that they depend on all of them, but not specifically on anyone of them), but depend specifically on the roles played by their members (or, better, on the concepts that classify their members). This means that they also indirectly depend on descriptions.

In [3] many different kinds of collectives have been characterized, based on degrees of agreement, devisal, transparency, control and structure, but for the present purposes we can consider an undifferentiated notion of collective, which is exemplified equally well by a group of people running all together toward a shelter during a sudden tempest, by a group of fans performing the "ola" at the stadium, and by the employees of an enterprise.

All these notions are embedded in rich axiomatizations and presented in detail in [1], [2] and [3] and for them we refer to those papers. In the current analysis we are just interested in using them as bases upon which to build a preliminary foundational analysis of the main entities and relations of an ontology of organizations.

III. Our Building Blocks

So far we have presented those notions that have already been dealt with in papers written by people of our laboratory (LOA). In the following we'll try to single out which are the main entities of an ontology of organizations, which are the connections between these entities and the others previously presented, which are the peculiar properties they acquire for the fact of being embedded in an organizational setting and the relations they entertain with each other.

The entities that populate the organizational settings are: organizations themselves, the agents who are member of the organization and who can act in it and sometimes for it, the roles that these agents play, other "organizational concepts", namely concepts that are expressly created for being used inside the organizational setting and, finally, norms and descriptions; they can define and constitute organizations themselves, they can define the concepts used inside organizations and can regulate the behavior of agents and organizations.

For what concerns agents, a couple of works ([6] and [3]) have been dedicated to the analysis of their features based on their mental attitudes, plans and goals, but these are just preliminary inquiries and they can be ignored for the sake of simplicity in this work, since at this stage we are only interested in the capability they have of acting on behalf of organizations, in virtue of some roles they play inside those organizations.

[2] A detailed axiomatization of descriptions is given in [4] and [5].

[3] Once we give a formal account, this allows us to express both social concepts and ground objects in first order language (see [2]).

[4] People often put both these classes of objects in the same domain of discourse when engaged in a conversation.

Something that is for sure of extreme interest for an ontological account of organizations is a study of the notion of collective intentionality and collective attitude in general: are these the product or the sum of the individual attitudes of the agents composing the collective, or are these some kind of primitive notions, that are not directly a consequence of these individual attitudes?

A last thing that is important to notice and that holds for all these categories is that organizations, social roles and concepts and norms are all social objects and, hence, non physical entities. There have been many debates around the physical character of social objects and the literature presents a lot of controversial issues (see [7], [8] and [9]), but a couple of examples can illustrate why we decided to take the non physical stance.

First of all, if a person is judged guilty of a serious crime, (s)he can be arrested and imprisoned; conversely, it is not possible to put to jail a company, like FIAT. For roles the language is less clear, in the sense that at a first glance it seems possible to arrest the President of FIAT, but in this case the police is not really arresting the President, rather the person that in that specific moment is playing the role of President.

Maybe a more evident example is that of hitting: while it is possible to hit a person, a building or a book, it sounds rather odd to say that I've taken a stick and I have furiously hit an organization, a role, a concept or a rule.

.1 Organizations

Organizations are obviously the main subject of our analysis. At least if we use the term with its classical meaning, they are complex social entities that are created and sustained by human agents[5]. A bit more specifically, an organization is a complex entity linked to a group of people that are thus able to constitute and regulate complex activities that otherwise could not be accomplished by non coordinated individuals.

With respect to the ontological nature of organizations, we can say that the literature has developed mainly around three fundamental questions:

- Are organizations social groups or different kinds of entities?
- Are organizations agents? If this is the case, which kind of agents are they?
- Do they keep their identity through time and changes? How?

With respect to the first question, in general in literature organizations are considered as distinct from social groups, based on the fact that normally social groups are thought of as sets of people connected by some kind of tie and conscious of this tie. On the other hand, at least intuitively, the word "organization" recalls some organized structures where knowledge is heterogeneously distributed, so that some members can be unaware of the tie that links them to people they can even ignore the existence of [10].

As for the second question, this constitutes the main subject of the literature on organizations in legal and moral philosophy, where it raises fundamental issues as personhood and responsibility of organizations. There's a fairly wide agreement on the fact that organizations have a personality and identity of their own and thus they are agentive entities ([11], [12]), but they act in a very peculiar way, namely through the actions of some agents who, in virtue of the roles they play, are delegated to act on their behalf[6]. Not only this: their actions (the actions these agents perform on their behalf) are of a particular form, that we can call "institutional". The President doesn't hit a piece of wood with a stick on behalf of the organization he's president of (unless this is a symbolic gesture with some further meaning), but he can very easily sign a contract on behalf of it. In other terms, every act which is indirectly performed by an organization must be institutional.

The third question has instead been answered by claiming a sort of "immortality" of organizations with respect to their members, in the sense that they preserve their identities through the turnover of people occupying roles ([8], [13]) and positions in it and they can even survive to the elimination of some of their constituent roles.

Our hypothesis is that organizations are social individuals; differently from social concepts and roles, they don't classify particulars (like agents or physical objects). They are agents, so they can create new norms, can play roles and can act by means of some member agents who play particular roles inside it.

Differently said, using [3]'s terminology, they depute their actions to some roles, which in turn classify individual agents, who are the ones that ultimately act.

.2 Roles and Concepts

Social roles and social concepts have already been described and analyzed at length in [3] and especially in [2], but here we'll mainly concentrate on those roles that classify intentional agents and social concepts that classify non agentive physical objects (like inanimate things).

Starting from roles, we can sum up their main features in the following way. First of all, a role can be played by different entities, at different times or even simultaneously; conversely, the same entity can play different roles, even simultaneously, so there's no necessary relation between a role and its player(s), so an entity can change role and also play the same role more than once. Roles are intrinsically relational, in the sense that, at a definitional level, they depend on the definition of other roles; a definition of a role cannot be given "in isolation" (let's think about the roles employer/employee, buyer/seller...). Finally, they are linked to some specific kinds of entities that provide explicit definitions for them; in the case of organizations, we can think about these entities as norms and descriptions.

Roles are also attached to an unusual notion of agentivity: they cannot act themselves, but they classify entities (like intentional agents) who can act[7].

In [2] some relations between roles are also analyzed. For instance, a role can specialize another role, as in the case of

[5] Nowadays many researches in the Artificial Intelligence domain are focused on the creation of "artificial agents' societies".

[6] We refer to the section on Agentive Figures of [3] for a deeper explanation of the relations of *deputing* and *acting for* holding between organizations and roles and organizations and agents playing those roles respectively.

[7] Sometimes it is common to say that someone acted in a certain way because (s)he was acting as the President of a certain organization. A possible way to deal with such kinds of expressions is to introduce a new kind of entity in the ontology that we could call *qua-entity*. Some discussions on this issue are presented in [2] and, more extensively, in [14].

"Italian Prime Minister", which is a specialization of the role "Prime Minister": some agent is Prime Minister because in particular (s)he is Italian Prime Minister. More interesting for our purposes is the relation that has been called *requirement*: it can be required that an agent, in order to assume a role, must have previously assumed another role. Again with Italian Prime Minister: in order to play the role of Italian Prime Minister, an agent needs to have previously played (and in this case (s)he must also still play) the role of Italian citizen.

This relation is very interesting because often in organizations there is a precise hierarchy of roles and there is a kind of "forced path" to follow in order to reach a certain position and play a determinate role.

Finally, the importance of the notion of social role or, more generally, of social concept in organizations is not only relevant for the case of agents, but also for non agentive objects. As a matter of fact, organizations have the capability of ascribing a certain *status* to certain objects: for instance, a piece of paper can acquire the status of bill or receipt because there's an organization whose members, if some norms are respected, recognize it as such.

Here we come to the third important building block for an ontology of organizations: descriptions and norms.

.3 Descriptions and Norms

In our account, all norms are descriptions. So, in a sense, they constitute the context inside of which both organizations and their members are defined.

This is in our opinion a very important part of the ontology of organizations that has not yet been addressed satisfactorily. So, we start here an informal analysis with the aim of giving a conceptual clarification of the issue as a starting point for a later formal analysis.

Following the literature (taking inspiration mainly from [15], [16] and [17]), we have singled out three different kinds of norms; the distinction is based on the different functions they have.

1. *Constitutive Norms*: they have a defining function: they create new concepts, roles, social individuals; they can also establish which are the requirements that an entity should meet in order to be classified under a certain role or concept.
2. *Deontic Norms*: they regulate the behavior of social entities: what they are allowed to do (directly or indirectly), what they are obliged to do etc. They create constraints on these behaviors inside organizations. In particular, they regulate the behavior that agents must observe when they play determinate roles. There are also deontic aspects connected with non agentive social concepts: for instance, the possession of a certain object that has acquired a social status can testify the fact that the owner of that object has the permission or the prohibition to do something (think about legal documents).
3. *Technical Norms*: they describe the correct procedure to do something [18]. Their social status comes from the fact that they are also created and accepted by communities of agents and, similarly as deontic norms, they also have the purpose of constraining the behavior of certain members of the organization, but they are distinguished by the fact that they are not "assertory" (you must do this and that), but are like suggestions. They are often used in organizations and they are very useful[8].

IV. Basic Relations

After having presented the building blocks of our framework, we start analyzing the relations that bind together these blocks. In this section we consider two basic relations for organizations, the validity relation and the representation relation. Before providing some intuitions about them, we must say that both relations need to be specifically considered in the institutional framework we are working on, and not in a wider sense. Therefore, the validity relation has to be seen as an institutional relation that holds between norms and organizations and not as a logical notion. The same is true for the representation relation, a relation holding among agents, that has nothing to do with the notion of representation dealt with in philosophy of mind. Another remark is important: as we shall see, the validity relation and the representation relation are respectively linked to the commitment and the delegation relations. In a sense, we can say that these latter notions are "more fundamental" than the former. They are not specific relations concerning only organizational settings, but rather very basic relations that characterize the whole social environment and are not limited to institutional aspects; surely they deserve a deeper and separate analysis.

A. Validity

What does it mean for a norm to be valid? There are well known problems related to the notion of validity in the literature of the modern theory of law, and many different answers have been given to them at least by [19], [20] and [18]. We do not enter in these details here, following our goal to give a general framework for organizations, but some intuitions on this basic notions are needed.

As we stated before, a (complex) description defines an organization. In this description there is all that is required to specify what the organization is, from its general purposes (making money or the revolution, for instance) to its concepts and roles (president, CEO, comrade etc.), and to the deontic and technical norms that the players of some role defined in it must follow.

We believe that this is not enough. We need something more than an abstract specification of what this social object (organization) is: we need another relation between the description and the organization. We will call this validity relation. We believe that this notion of validity is linked with the dimension of social commitment, i.e. it is something that turns the description into a prescription for agents. When we consider the description that defines the concept *triangle*, we are in no way "legally forced" by this description, and in the same way a theory that simply defines an organization has no legal power for the agents related to it. Therefore, a description is valid when a particular

[8] A last distinction that could be made about norms is based on their origin. Either norms are institutionally created by an authority and thus explicitly encoded on some physical support, or they can emerge from social practices. In this latter case they can be respected and still remain implicit, or they can later evolve in institutional, when their usefulness is recognized and someone in the organization decides to encode them.

social event occurs. This social event (take for instance a poll, some official publication, a promise and so on) creates a social commitment among the agents related to the organization. This relation is exactly what makes the difference between simple descriptions and (systems of) norms: norms are those descriptions that are valid within and for an organization.

With this relation of validity we can define also the relations of institutionalization and affiliation. Intuitively, "being institutionalized", for a role or, more generally, for a concept means to be embedded in the structure of the organization. Like the validity relation for norms, it is used to give a "legal status" to concepts and roles that are used and structured in the organization. On the other hand, the relation of affiliation indicates the conditions under which agents are member of organizations. For instance, an individual that plays the role of researcher is affiliated to a University and his/her role is institutionalized in the University.

B. Representation

Another important relation that we take into account is the representation relation. This relation holds between agents. As we stated before, this relation is linked to the delegation relation. In Castelfranchi's view [21]:

> [..] in delegation an agent A needs or likes an action of another agent B and includes it in its own plan. In other words, A is trying to achieve some of its goals through B's behaviours or actions; thus A has the goal that B performs a given action/behaviour.

This important relation holds in many different social contexts and, among these, also in the institutional one, but it is not specific of it. The relation that characterizes the institutional and organizational contexts and is peculiar of them is the relation of representation.

In our remarks on the nature of organizations we pointed out their immateriality and their agentivity as fundamental properties, but then a problem arises: how can a non physical object act? Partially following [22] and [11] we suppose that there is one (or some) relevant agent(s) of the organization (for example the founder) that gives the authority to one (or some) other agent(s) to act on behalf of the organization. In this way any action that has an 'institutional meaning' and is performed by the "delegate" agent could be seen as performed by the organization itself. Therefore, in our view, the relevant agent(s) (i.e. the founder of the organization) must have established in the normative system of the organization this capability of the agents of acting on behalf of it.

This could be done, in our framework, by means of the representation relation. Generally speaking, the representation relation is a delegation relation that holds between agents that are classified by two roles: the *representative* and the *represented* role. Differently from the delegation relation, if the representation relation holds, the delegant cannot perform him/herself the action that (s)he wants or needs the delegate to do. The case of organizations is clearly one of these. Organizations, as immaterial entities, cannot act without a physical agent who acts for them.

Therefore, any organization has at least a representative role and a represented role defined in its normative system. The represented role must classify the organization itself and the representative role must classify at least another role defined by the normative system of the organization, for example the role "President". The representative role must, for the aforementioned reasons, classify a role, like "President", that, in turn, classifies only agentive physical objects. These can be seen as necessary conditions in order for the rappresentation relation to hold.

V. Formal Characterization

In this section we will provide a first draft of a formal characterization in first order logic of the main notions and relations presented in the paper. In order to do that, we need to informally introduce some predicates of DOLCE and to use some of the axioms and formulas previously presented in [2][9].

The predicates of DOLCE we will refer to are:

- $ED(x)$ standing for "x *is an endurant*", i.e., an entity that is *wholly* present at any time it is present, e.g., a car, Berlusconi, K2, a law, some gold...;
- $PD(x)$ standing for "x *is a perdurant*", i.e., an entity that is only partially present, in the sense that some of its temporal parts may be not present, e.g., reaching the summit of K2, a conference, eating, being open...;
- $SOB(x)$ standing for "x *is a social object*", i.e., an endurant that: (i) is not directly located in space and, in general, has no direct spatial qualities; (ii) depends on a community of intentional agents, e.g., a law, an economic system...;
- $ASO(x)$ standing for "x *is an agentive social object*", i.e., a social object that has, in some sense, intentionality, e.g., the Italian Republic...;
- $NASO(x)$ standing for "x *is a non-agentive social object*", i.e., a social object that has no intentionality, e.g., a currency...;
- $TL(x)$ standing for "x *is a temporal location*", i.e., a temporal interval or instant;
- $\mathsf{PC}(x,y,t)$ standing for "*the endurant* x *participates in the perdurant* y *at time* t", i.e., a person who participates in a discussion.

The next step is that of taking the notions of concept (CN) and description (DS) together with some of the relations holding among them from [2].

First we introduce restrictions on arguments for concepts and descriptions:

(KA1) $DS(x) \rightarrow NASO(x)$
(KA2) $CN(x) \rightarrow NASO(x)$
(KA3) $DS(x) \rightarrow \neg CN(x)$

Then, we reuse some of the main axioms, modified as for including in the formalization the notion of social individual (SI) that in [2] was only informally introduced:

(A1) $SI(x) \rightarrow ASO(x)$

A social individual is an agentive social object; examples of social individuals are the MILAN football club and the Italian

[9] From a notational standpoint, axioms, definitions and theorems imported from [2] can be distinguished from the ones that are originally introduced in the paper by the fact that they are preceded by a K letter.

Presidency.

(KA4) $\mathsf{US}(x,y) \rightarrow (CN(x) \wedge DS(y))$

This axiom is an argument restriction on the US relation, which can range only over concepts and descriptions. The intuitive meaning of the axiom is that a concept is used in a description. We want to apply this axiom also to social individuals, thus we modify it in this way:

(A2) $\mathsf{US}(x,y) \rightarrow ((CN(x) \vee SI(x)) \wedge DS(y))$

So, the US relation holds also between social individuals and descriptions.

(KA5) $\mathsf{DF}(x,y) \rightarrow \mathsf{US}(x,y)$

This states that the definition (DF) relation is a specialization of the use (US) relation and that concepts and social individuals are defined by descriptions.

(KA6) $CN(x) \rightarrow \exists y(\mathsf{DF}(x,y))$

This axiom states that every concepts must be defined by at least a description. Even in this case, we want to apply the axiom also to social individuals:

(A3) $(CN(x) \vee SI(x)) \rightarrow \exists y(\mathsf{DF}(x,y))$

(KT1) $\mathsf{DF}(x,y) \rightarrow (CN(x) \wedge DS(y))$

Thus, the theorem above is no more valid and the theorem below follows from (A2) and (KA5):

(T1) $\mathsf{DF}(x,y) \rightarrow ((CN(x) \vee SI(x)) \wedge DS(y))$

Finally, in the following we will use the notion of classification (CF), that we will also import.

(KA11) $\mathsf{CF}(x,y,t) \rightarrow (ED(x) \wedge CN(y) \wedge TL(t))$

Now, some new notions are introduced. First of all, for the sake of simplicity, we introduce the predicate Agent (AG), that is the union of the categories of APO and ASO:

(A4) $AG(x) \rightarrow (APO(x) \vee ASO(x))$

We introduce the notion of social event (SEV), which is a particular kind of perdurant:

(A5) $SEV(x) \rightarrow PD(x)$

A further characterization of social event is the following:

(A6) $SEV(x) \rightarrow \exists y,z(AG(y) \wedge SOB(z) \wedge \mathsf{PC}(y,x,t) \wedge \mathsf{PC}(z,x,t))$

(A6) tries to capture the intuition that a social event is an event in which participate both (at least) an agent and a social object. For instance, a social event, like a poll, involves agents and social objects like parties and ballots. We have decided to use a single variable for time for simplicity, thus assuming that agents and social objects participate both for the whole duration of the event[10].

[10] We are aware of the fact that this is not obvious, but it shouldn't be too difficult to distinguish the time of participation of the agent and the time of participation of the social object and to characterize the relations holding between these two time periods.

(A7) $\mathsf{VAL}(x,y) \rightarrow SI(y) \wedge \mathsf{DF}(y,x) \wedge \exists z(SEV(z) \wedge \mathsf{PC}(x,z,t) \wedge \mathsf{PC}(y,z,t))$

Here we introduce a new primitive, validity (VAL) and (A7) explains that, in order for a description to be valid for a social individual, a necessary condition is the occurrence of a social event in which both the social individual and the description participate[11].

(D1) $\mathsf{INST}(x,y) \triangleq CN(x) \wedge \exists z(\mathsf{VAL}(z,y) \wedge \mathsf{US}(x,z))$

(D1) defines the relation, called institutionalization (INST), between a concept and a social individual when such a concept is used by a description that is valid for the social individual.

(A8) $RL(x) \rightarrow CN(x)$

In [2] a precise definition of roles (RL) is given, to which we refer. Here it is sufficient to point that roles are concepts.

(D2) $\mathsf{AFF}(x,y,t) \triangleq AG(x) \wedge \exists z(RL(z) \wedge \mathsf{CF}(x,z,t) \wedge \mathsf{INST}(z,y))$

(D2) defines the relation, called affiliation (AFF), between an agent and a social individual in a certain time interval. An agent is affiliated to a social individual iff (s)he plays a role that is institutionalized for the social individual.

(A9) $ORG(x) \rightarrow \exists y \mathsf{AFF}(y,x,t)$

With this machinery we can say that a necessary condition for a social individual to be an organization (ORG) is the existence of at least one agent who is affiliated to it.

From (A9), (D1) and (A7), it follows:

(A10) $ORG(x) \rightarrow SI(x)$

all organizations are social individuals.

This is only a preliminary characterization, in order to have a formal definition of organizations as described above, we need to characterize the representation (REP) relation just described. Thanks to the REP relation, (A10) and (A9) could be replaced by the following definition:

$ORG(x) \triangleq \exists y,z(\mathsf{AFF}(y,x,t) \wedge \mathsf{REP}(z,x))$

In order to illustrate our main entities and relations, let us consider an example (illustrated in figure 1) in the context of our formal framework. The individual Carlo Azeglio Ciampi is classified by the role President of Italy. This role and the organization Italian State are defined by the Italian Constitution, that is a description. Moreover, the role President of Italy is institutionalized by the Italian state and, because of this, Ciampi (as individual) is affiliated to the Italian State. Finally, the Italian Constitution itself is valid for the Italian State.

In figure 1, as in [2], the following conventions are assumed:

- universals (predicates) are represented in italics, with first capital letter;
- individuals (instances) are represented in type with small letters;

[11] The intuition underlying this definition of validity is that during a social event, a link is established between an institution and the description and norms that define it, thus all these elements must participate to the social event.

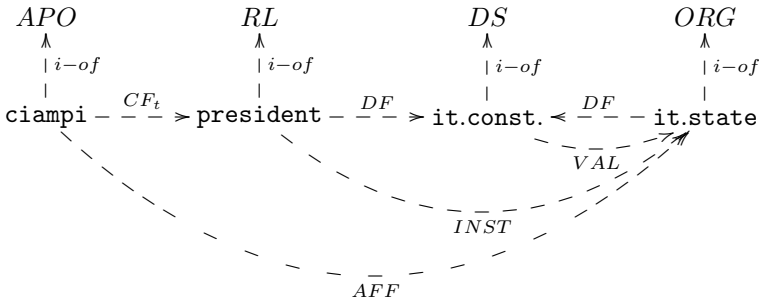


Fig. 1. Main relations and entities illustrated by the Ciampi example.

- relations between individuals are represented by dashed labeled arrows:
  $\mathtt{a} - \overset{R}{-} \succ \mathtt{b}$ stands for: $R(\mathtt{a}, \mathtt{b})$;
- the "instance-of" relation between a particular and a universal is labelled by $i - of$.

VI. Related Works

As far as we know, there are not so many works on the ontology of organizations. Those available can be divided according to the different perspective they take.

Most of the philosophical studies on organizations concentrate on ethical issues, like moral personhood and responsibility ([23]) and very few of them have a formal flavor. An important exception is the account given by Raimo Tuomela. His analysis of organizations in [17] is part of a wider project about institutional reality, strongly based on the analysis of the notion of collective intentionality, joint actions and social practices.

The notion of normative system is also analyzed but, differently from our paper, this is done by looking at the dynamics, trying to understand – for instance – which actions are the agents in the organization allowed or not allowed to do.

On the other hand, in computer science some works on the ontology of organizations can be found, like [24], [25], [26], [27], [28], even though most of them are really works of enterprise modeling. If we consider enterprises as a special kind of organizations, these works can be seen as more specifically oriented than ours, which is instead more "top-level". As a consequence of this specificity, they mainly focus on workflow, activities, time-constrained processes and all those elements relative to the dynamics of organizations, thus resulting in ontologies of action.

Another relevant difference of all these approaches with respect to ours is that their scope is much wider, in the sense that they try to be global in considering not only structural aspects, but also teleological aspects, interaction patterns, and many more primitive entities. On the other hand, even if most of them represent in their frameworks some of the relations that we have concentrated on in the paper (like institutionalization, affiliation etc.), they treat them as "black boxes", while we try to "look inside the boxes". In our opinion this is something that has to be done in order to better understand what these basic relations are and to be able to build upon them.

Probably the main reason of these differences is to be imputed to the fact that often these works move from the needs that emerge in applications and try to give a theory that deals satisfactorily with these problems, while we try to reach first a "clean" theoretical account and then we try to apply it to concrete scenarios.

VII. Future Work

This paper is meant to be a prosecution of some previous works on the social dimension of the ontology DOLCE and is mainly an attempt to present the basic entities and relations of the domain of organizations, which is included in the social realm. As a further step, we want to improve this preliminary work in four directions, starting from the two just sketched relations.

1. As a first move, we'll try to clearly link the notion of rapresentation with the notion of *qua-individual*. As shown in [14], if a classification relation holds between a role and an endurant, a third entity "arises": a qua-individual. As an example, take the situation in which Ciampi, an agentive physical object, is the President of the Italian Republic, i.e. is classified by this role. For the whole time span in which this relation holds an entity, a qua-individual (namely, Ciampi *qua-President-of-Italy*), exists. In [14] we hold that qua-individuals actually participate in events. Following the example, the Italian constitution – i.e. the normative system of the Italian State – states that "the president may dissolve one or both chambers after having consulted their speakers". Therefore, when Ciampi dissolves the chambers *qua-President-of-Italy*, it is natural to hold that it is the qua-individual Ciampi *qua-President-of-Italy* who performs the action. But the qua-individual performs the action also as a rapresentative of the Italian State, so there is a sense in which it is the Italian State that dissolves the chambers. If so, how many individuals participate in this action? Who is, ultimately, the agent which performs the action? Which are the relations between these entities? Representation and qua-individuals seem to be somehow linked, so we have to inquire the nature of this link.
2. A second possible improvement is to link the affiliation with the representation relation. In order to understand this complex link, we need to make a comparison between the *acting for* relation (between agents and organizations) and the *membership* relation (between agents and collections) developed in [3] with our affiliation and representation relations. Moreover, we need to investigate if the elements we have considered in the paper are enough in order to de-

fine this relation.

3. Thirdly, organizations are composed by human agents, but also by pluralities of non agentive entities. So, as mentioned in section II, the notions of collection and collective are central.

   In [3] collections are considered to be social objects that (generically) depend on their members; consider, for instance, a collection of books in a library, suppose the collection of books of the Library of Congress, which remains the same entity even if some books are lost and others acquired over time. If we consider the Library of Congress as an organization, we could call the collection of its books as one of its "resources" (aside with others, like its furniture, buildings and so on). We could also say that for a collection, in order to be a resource for an organization, it must have at least one role defined in the normative system of the organization itself. Let's then recall the main difference between collections and collectives: members of the latter are agents. So, similarly, we could consider the staff of the Library of Congress as a collection where the roles that characterize it are defined in the normative system of the Library.

   The idea is that we can consider the notions of resources and staff of an organization as a specialization of the notions of collection and collective and thus try to reuse some of the analyses already done for these two latter notions.

4. Finally, in this paper we have tried to investigate some features of organizations by considering them in isolation. This was done just for simplicity reasons and we are well aware of the fact that a complete account would require an analysis of multiple organizations interacting in a wider environment. A special case would be that of organizations that are embedded in other, bigger, organizations. As an example, consider the relation between a University, suppose the University of Trento and one of its Departments, for instance the Philosophy Department. We could say that the latter is "contained" in the former, but what does it mean? What is required for this relation to hold? What happens to the normative systems of both these social individuals? Must there be some special roles defined into their normative systems?

These are some of the questions that are left unanswered in this paper, but that can help to enhance the understanding of what is the ontological nature of organizations.

ACKNOWLEDGMENTS

We would like to thank Claudio Masolo and Robert Trypuz for the fruitful feedbacks and discussions. We are also indebted to Giancarlo Guizzardi for some improvements in the camera-ready version.

REFERENCES

[1] Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino, and Alessandro Oltramari, "Wonderweb deliverable d18," Tech. Rep., 2003.

[2] Claudio Masolo, Laure Vieu, Emanuele Bottazzi, Carola Catenacci, Roberta Ferrario, Aldo Gangemi, and Nicola Guarino, "Social roles and their descriptions," in *Ninth International Conference on the Principles of Knowledge Representation and Reasoning*, Whistler Canada, 2004, Accepted.

[3] Emanuele Bottazzi, Carola Catenacci, Aldo Gangemi, and Jos Lehmann, "From collective intentionality to intentional collectives: An ontological perspective," *Cognitive Systems Research (submitted)*, 2006.

[4] Aldo Gangemi and Peter Mika, "Understanding the semantic web through descriptions and situations," in *International Conference on Ontologies, Databases and Applications of Semantics (ODBASE 2003)*, Robert et al. Meersman, Ed. 2003, Springer Verlag.

[5] A. Gangemi, Carola Catenacci, J. Lehmann, and S. Borgo, "Task taxonomies for knowledge content," Tech. Rep., EU 6FP METOKIS Project D07, http://metokis.salzburgresearch.at, 2004.

[6] Roberta Ferrario and Alessandro Oltramari, "Towards a computational ontology of mind," in *Formal Ontology in Information Systems, Proceedigs of the Intl. Conf. FOIS 2004*, Achille C. Varzi and Laure Vieu, Eds. 2004, pp. 287–297, IOS Press.

[7] Adolf Reinach, "The apriori foundations of civil law," *Aletheia*, vol. III, pp. 1–142, 1988.

[8] Barry Smith, "Social objects," http://ontology.buffalo.edu/socobj.htm, 2002.

[9] Giuseppe Lorini, *Dimensioni giuridiche dell'istituzionale*, Cedam, Padova, 2000.

[10] Margaret Gilbert, *Social Facts*, Princeton University Press, Princeton, New Jersey, 1992.

[11] Jean-Jacques Rousseau, *The Social Contract*, Oxford University Press, Oxford, UK, 1997/1762.

[12] Maurice Hauriou, *Teoria dell'istituzione e della fondazione*, Giuffrè, Milano, 1967.

[13] John Ladd, "Morality and the ideal of rationality in formal organizations," *The Monist*, vol. 54, no. 4, pp. 488–516, 1970.

[14] Emanuele Bottazzi, Roberta Ferrario, Giancarlo Guizzardi, Claudio Masolo, and Laure Vieu, "Relational roles and qua-individuals," *paper accepted for the AAAI Fall Symposium on Roles, an interdisciplinary perspective, November 3-6, 2005, Hyatt Crystal City, Arlington, Virginia*, 2005.

[15] John R. Searle, *The Construction of Social Reality*, The Free Press, New York, 1995.

[16] Raimo Tuomela and Maj Bonnevier-Tuomela, "Norms ad agreement," *European Journal of Law, Philosophy and Computer Science*, vol. 5, pp. 41–46, 1995.

[17] Raimo Tuomela, *The Philosophy of Social Practices*, Cambridge University Press, Cambridge, UK, 2002.

[18] G. H. von Wright, *Norm and action : a logical enquiry*, International library of philosophy and scientific method. Routledge and Kegan Paul, London, 1963.

[19] H. L. A. Hart, *The concept of law*, Clarendon law series. Clarendon Press, Oxford, 1961, by H.L.A. Hart. 23 cm.

[20] Hans Kelsen, *Pure theory of law*, California library reprint series. University of California Press, Berkeley, california library reprint series edition edition, 1967, by Hans Kelsen ; translation from the second (revised and enlarged) German edition by Max Knight. 24 cm.

[21] Cristiano Castelfranchi, "Grounding we-intention in individual social attitudes: On social commitment again," in *Realism in Action - Essays in the Philosophy of Social Sciences*, M. Sintonen and Kaarlo Miller, Eds. Kluwer, Dordrecht, 2003.

[22] Thomas Hobbes, *Leviathan*, OUP, Oxford, 1996.

[23] Peter A. French, *Collective and Corporate Responsibility*, Columbia University Press, 1984.

[24] Mark S. Fox, Mihai Barbuceanu, Michael Gruninger, and Jinxin Lin, "An organisation ontology for enterprise modelling," in *Simulating Organizations: Computational Models of Institutions and Groups*, K. Carley and L. Gasser, Eds., pp. 131–152. AAAI/MIT Press, Menlo Park, CA, 1997.

[25] Michael Gruninger and Mark S. Fox, "The logic of enterprise modelling," in *Modelling and Methodologies for Enterprise Integration*, Bernus P. and Nemes L., Eds. Chapman and Hall, 1996.

[26] Virginia Dignum, *A Model for Organizational Interaction: based on Agents, founded in Logic*, Ph.D. thesis, Universiteit Utrecht, 2004.

[27] Jan Dietz, "The atoms, molecules and fibers of organizations," *Data and Knowledge Engineering*, vol. 47, pp. 301–325, 2003.

[28] Mike Uschold, Martin King, Stuart Moralee, and Yannis Zorgios, "The enterprise ontology," *The Knowledge Engineering Review*, vol. 13, no. 1, pp. 31–89, 1998.

A Context-Based Enterprise Ontology

Mauri Leppänen
Department of Computer Science and Information Systems
P.O. Box 35 (Agora), FI-40014 University of Jyväskylä, Finland
mauri@cs.jyu.fi

Abstract

The main purpose of an enterprise ontology is to promote the common understanding between people across different enterprises. It serves also as a communication medium between people and applications, and between different applications. This paper outlines a top-level ontology, called the context-based enterprise ontology, which aims to promote the understanding of the nature, purposes and meanings of things in enterprises with providing basic concepts for conceiving, structuring and representing things within contexts and/or as contexts. The ontology is based on the contextual approach according to which a context involves seven domains: purpose, actor, action, object, facility, location, and time. The concepts in the ontology are defined in English and presented in meta models in a UML-based ontology engineering language.

1. Introduction

Numerous applications are run in enterprises to provide information for, and to enable communication between, various stakeholders, inside and outside the organization. Currently, an increasingly large portion of enterprise knowledge is hold, processed and distributed by applications. Enterprise knowledge is "local knowledge" by its nature, in that its meaning and representation is agreed in relatively small, local contexts. A prerequisite for the successful use of applications is, however, that the common understanding about that knowledge is reached and maintained across the enterprise(s). Especially in modern inter- and intra-organizational applications the need to support the understanding of shared knowledge is crucial [2]. This implies that besides technical interoperability, the enterprises are facing with the challenge of achieving semantic and pragmatic interoperability among the applications.

For human beings to understand what individual things in reality mean they need to know what purposes the things are intended for, by whom, when, and where, how they are related to other things and environment, how they have been emerged, created, and/or evolved, when and where, etc. Shortly, they need to know about contexts where the things appear, have appeared, and/or are to be appeared, and also about the things related to them in those contexts. Considering this, it is understandable that context plays an important role in many disciplines, such as in formal logic, knowledge representation and reasoning, machine learning, pragmatics, computational linguistics, sociolinguistics, organizational theory, sociology, and cognitive psychology. In most of these fields, the notion is used, in particular, to specify, interpret, and infer meanings of things through the knowledge about the contexts they appear.

In the recent years a number of enterprise and business ontologies and frameworks (e.g. [8], [38], [25], [9]) have been proposed. Some of them are generic, whereas the others are aimed at specific business fields (e.g. UNSPC, NAICS, and OntoWeb for e-commerce). In addition, there are several enterprise modeling languages (e.g. IEM, EEML, GRAI/Actigrams). The main purpose of an enterprise ontology is to promote the common understanding between people across different enterprises. It serves also as a communication medium between people and applications, and between different applications. Taking into account the significance that the sharing of meanings has in communication within enterprises as well as experiences got from the use of context in capturing meanings in other disciplines, it is surprising how ignored a contextual view is in current enterprise ontologies. We propose that the semantic and pragmatic interoperability of applications in enterprises should be advanced by the more explicit use of context and other contextual concepts in enterprise ontologies.

Our aim in this study is to present a context-based enterprise ontology. It is a top-level ontology [11], which provides a unified view of the enterprise as an

aggregate of contexts. This ontology can be specialized into task ontologies or domain ontologies to meet special needs of the enterprise, but still maintaining connections of the specialized things to their contexts. The concepts in the context-based enterprise ontology are defined in English and presented in meta models in a UML-based ontology representation language. The UML language has been adopted as the basis because it has a very large and rapidly expanding user community, it is supported by widely adopted engineering tools, and there are positive experiences from the use of UML in presenting ontologies (e.g. [5], [39]). We apply a subset of the concepts of the class diagram.

The article is structured as follows. In Section 2 we will define the notion of context and the contextual approach, and describe the overall structure of the context-based enterprise ontology. In Section 3 we will define the contextual concepts of the ontology and present them in meta models. We will end with the summary and conclusions.

2. Context and Contextual Approach

Based on a large literature review about the notion of context in several disciplines, we conclude that a *context* is a whole, composed of things connected to one another with contextual relationships. A thing gets its meaning through the relationships it has with the other things in that context.

To define a proper set of contextual concepts we draw upon relevant theories about meanings. Based on three topmost layers in the semiotic ladder [36], we identify semantics (e.g. case grammar [7]), pragmatics [22], and the activity theory [6], respectively, to be such theories. In semantics, context appears as a sentence context, in pragmatics as a conversation context, and in the activity theory as an action context.

Anchored on this groundwork and some "contextual" approaches (e.g. [35], [31], [27]), we define seven domains, which serve concepts for specifying and interpreting contextual phenomena. These contextual domains are: purpose, actor, action, object, facility, location, and time (Figure 1). Structuring the concepts within and between these domains is guided by the following scheme, called the seven S's scheme: *For **S**ome* purpose, ***S**omebody* does ***S**omething* for ***S**omeone*, with ***S**ome* means, ***S**ometimes* and ***S**omewhere*.

We define the *contextual approach* to be the approach according to which individual things are seen to play certain contextual roles in a context and/or to be contexts themselves. Following this approach, we define an *enterprise* to be an aggregate of contexts that are composed of people, information and technologies,

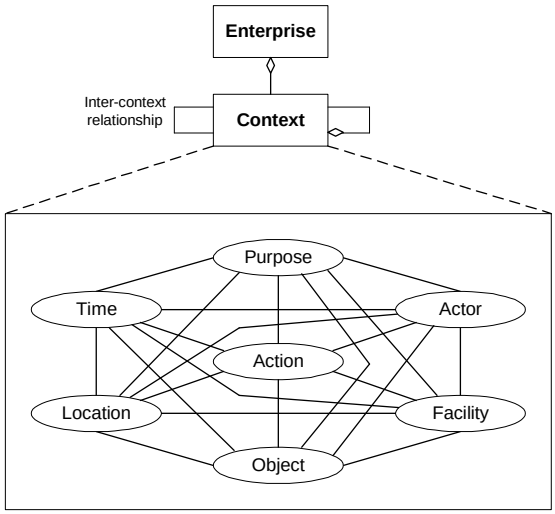


Figure 1. An overall structure of the context-based enterprise ontology

performing functions in a defined organizational structure, for agreed purposes, and responding to events, both internal and external, and needs of stakeholders. The contexts can be decomposed into more elementary contexts, and they are related to one another with inter-context relationships.

An ontology is an explicit specification of a conceptualization of some part of reality that is of interest [10]. The *context-based enterprise ontology* is an ontology which aims to promote the understanding of the nature, purposes, and meanings of the things in the enterprise with providing concepts and constructs for conceiving, structuring, and representing things within contexts, and/or as contexts. The ontology is intended to assist the acquisition, representation, and manipulation of enterprise knowledge via the provision of a consistent core of basic concepts and constructs.

In the next section we will first define the contextual domains and the most essential concepts within them. Due to the limitation of space, the location and time domains are excluded. In addition, we will shortly present relationships between the domains.

3. Contextual Domains

3.1 Purpose Domain

The *purpose domain* embraces all those concepts and constructs that refer to goals, motives, or intentions of someone or something (Figure 2). The concepts are also used to express reasons for which something exists or is done, made, used, etc. We use *purpose* as the general term in this domain.

A *goal* (of e.g. an actor or action) means a desired state of affairs ([25], [19]). It can also be related to an object, a facility, a location or a time (system), meaning the purpose, which they are aimed at. A *reason* is a basis or cause for some action, fact, event etc. [40]. It can be a requirement, a problem, a strength/weakness, or an opportunity/a threat. Between a goal and a reason there is the *dueTo relationship*, meaning that a reason gives an explanation, a justification or a basis for setting a goal.

We can specialize the goals based on their lifespan. *Strategic goals* are kinds of missions, answering questions such as "What is the direction of an enterprise in the future". Their spans are generally 5 – 10 years. *Tactic goals* show how to attain strategic goals. *Operative goals* are generally determined as concrete requirements that are to be fulfilled by a specified point of time. The goals can also be categorized based on whether it is possible to define clear-cut criteria for the assessment of the fulfillment of goals. *Hard goals* have pre-specified criteria, and *soft goals* have not [23].

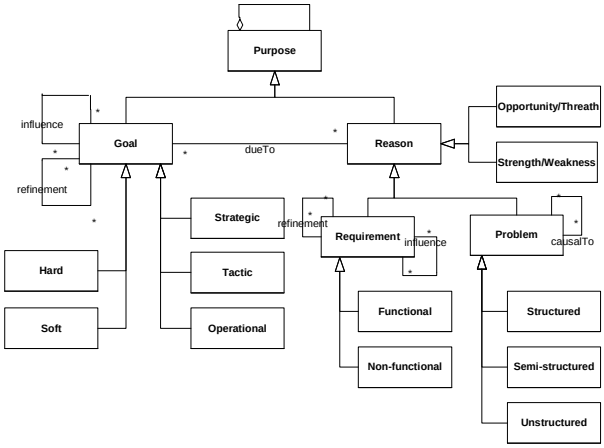


Figure 2. Purpose domain

Requirements mean something that are necessary and needed. They are statements about the future [28]. Actually, the goals and the requirements are two sides of a coin: some of the stated requirements can be accepted to be goals to which actors want to commit. A *functional requirement* can be achieved by performing a sequence of operations [20]. A *non-functional requirement* is defined in terms of constraints, to qualify the functional requirement related to it.

Instead of directly referring to a desirable state, a purpose can also be expressed through an indirect reference to problems that should be solved. A *problem* is the distance or a mismatch between the prevailing state and the state reflected by the goal [15]. To reach the goal, the distance should be eliminated or at least reduced. Associating the problems to the goals expresses reasons, or rationale, for decisions or actions towards the goals [30]. The problems are commonly divided into structured, semi-structured and unstructured problems [33]. *Structured problems* are those that are routine, and can be solved using standard solution techniques. *Semi-structured* and *unstructured problems* do not usually fit a standard mold, and are generally solved by examining different scenarios, and asking "what if" type questions.

Other expressions for the reasons, of not so concrete kind, are strengths, weaknesses, opportunities and threats related to something for which goals are set (cf. SWOT-analysis, e.g. [16]). *Strength* means something in which one is good, something that is regarded as an advantage and thus increasing the possibilities to gain something better. *Weakness* means something in which one is poor, something that could or should be improved or avoided. *Opportunity* is a situation or condition favorable for attainment of a goal [40]. *Threat* is a situation or condition that is a risk for attainment of a goal.

A general goal is refined into more concrete ones. The *refinement relationship* between the goals establishes goal hierarchies, in which a goal can be reached when the goals below it (so-called sub-goals) in the hierarchy are fulfilled (cf. [18]). The *influence relationship* indicates that the achievement of a goal has some influence, positive or negative, on the achievement of another goal (cf. [25], [18]).

As the goals and the requirements are two sides of a coin, the relationships between the requirements are similar to those between the goals. Consequently, a requirement can influence on another requirement, and a requirement can be a refinement of another requirement. The relationships between the problems manifest causality. The *causalTo relationship* between two problems means that the appearance of one problem is at least a partial reason for the occurrence of the other problem.

3.2 Actor Domain

The *actor domain* consists of all those concepts and constructs that refer to human and active parts in a context (Figure 3). Actors perform, own, communicate, borrow, send, receive etc. objects in the contexts. They are responsible for and/or responsive to triggering and causing changes in the states of objects in the same context, or in other contexts. We consider it important, from the philosophical viewpoint, to distinguish human actors from non-human actors, which are here regarded as tools (see Section 3.5).

An *actor* is a human actor or an administrative actor. A *human actor* is an individual person or a group of persons. A *person* is a human being, characterized by

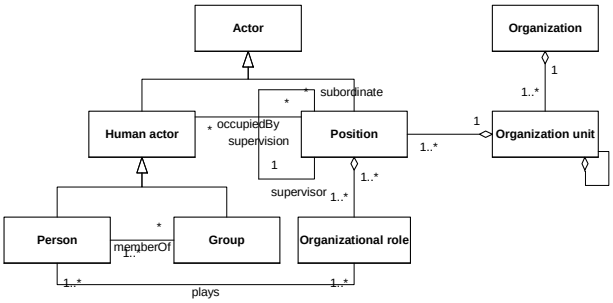


Figure 3. Actor domain

his/her desires, intentions, social relationships, and behavior patterns conditioned by his/her culture (cf. [3], [29]). A person may be a member of none or several *groups.* An administrative actor is a position or a set of positions. A *position* is a post of employment *occupied by* zero or many human actors. For each position, specific qualifications in terms of skills, demands on education and experience, etc. are specified.

An *organizational role*, shortly a role, is a collection of responsibilities, stipulated in an operational or structural manner. In the former case, a role is composed of tasks that a human actor occupying the position with that role has to perform. In the latter case, a role is charged with responsibilities for some objects. A role can be played by many persons, through or without the position(s) they hold.

The *supervision relationship* involves two positions in which one is a supervisor to another that is called a subordinate. A supervisor position has responsibility and authority to make decisions upon the positions subordinate to it, and those occupying the subordinate positions have responsibility for reporting on one's work and results to those occupying the supervisor position.

An *organization* is an administrative arrangement or structure established for some purposes, manifesting the division of labor into actions and the coordination of actions to accomplish the work. It can be permanent and formal, established with immutable regulations, procedures and rules. Or it may be temporally set up, like a project organization, for specific and often short-range purposes. An *organizational unit* is composed of positions with the established supervision relationships. An organization consists of organizational units.

3.3 Action Domain

The *action domain* comprises all those concepts and constructs that refer to deeds or events in a context (Figure 4). We use *action* as the generic concept to refer to things belonging to the action domain. Actions can be autonomous or cooperative. They can mean highly abstract work like studies in mathematics, or at the other extreme, physical execution of a step-by-step procedure with detailed routines.

There are a large number of action structures, which an action is a part of. We distinguish between the decomposition structure, the control structure, the temporal structure and the management – execution (Mgmt-Exec) structure.

In the *decomposition structure*, actions are divided into sub-actions, these further into sub-sub-actions, etc. Sub-actions may be functions, activities, tasks, operations, etc. Decomposition aims at reaching the level of elementary actions, where it is not possible or necessary to further decompose. The *control structure* indicates the way in which the actions are logically related to each other and the order in which they are to be executed. The control structures are: sequence, selection, and iteration. The *sequence relationship* between two actions act_1 and act_2 means that after selecting the action act_1 the action act_2 is next to be selected. The *selection relationship* means that after selecting the action act_1 there is a set of alternative actions act_2,.., act_n from which one action (or a certain number of actions) is to be selected. The *iteration relationship* means that after selecting the action act_1 the same action is selected once more. The selection is repeated until the stated conditions become true. The *temporal structure* is like the control structure but with temporal conditions and events.

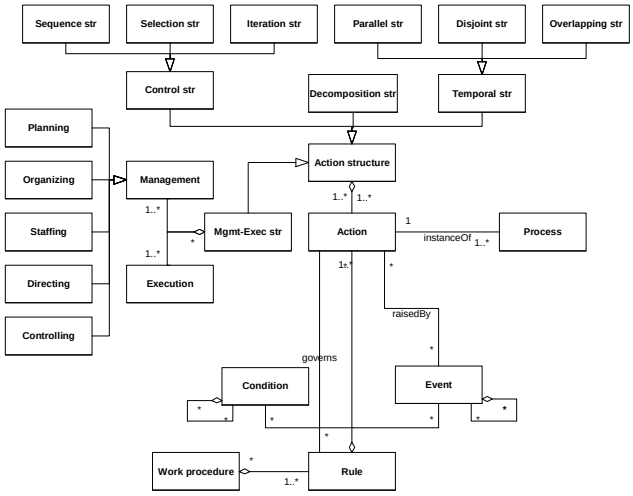


Figure 4. Action domain

The temporal structures are specified using temporal constructs, such as during, starts, finishes, before, overlaps, meets, and equal. Constructs are used to specify relationships between starting and/or ending events, or between durations of actions. With these

constructs, overlapping, parallel, disjoint (non-parallel) and overlapping executions of actions can be distinguished. Two actions are said to be *overlapping* if the durations of their executions overlap. The actions are (strictly) *parallel* if the durations are equal or the duration of one action is included in the duration of the other action. Two actions are said to be *disjoint* if their durations do not overlap.

The *management – execution structure* is composed of one or more management actions and those execution actions that implement prescriptions provided by the management actions (e.g. [26], [41], [14]). *Management actions* mean the *planning, organizing, staffing, directing and controlling* of execution actions, in order to ensure the achievement of goals and constraints (cf. [4], [34], [37]). The purpose of *execution actions* is to implement the prescriptions with the given resources.

The action structures are orthogonal to one another. This makes it easy to specialize the defined structures, and extend them with new ones, e.g. with the dichotomy of material and social actions (cf. speech acts [32]). The action structures are enforced by rules. A *rule* is a principle or regulation governing a conduct, action, procedure, arrangement, etc [40]. It is composed of four parts [12], event, condition, thenAction, and elseAction, structured in the ECAA structure. An *event* is an instantaneous happening in the context, with no duration. A *condition* is a prerequisite for triggering an action. A *thenAction* is an action that is done when the event occurs and if the condition is true. An *elseAction* is an action that is done when the event occurs but the condition is not true. An aggregate of related rules constitutes a *work procedure* (cf. [14]), which prescribes how the course of action should proceed. Depending on the knowledge of, and a variety of, actions, work procedures may be defined at different levels of detail [13]. An instance of an action is a *process*.

3.4 Object Domain

The *object domain* contains all those concepts and constructs that refer to something, which an action is directed to (Figure 5). It can be a message, a decision, an argumentation, a list of problems, a program code, a workstation, etc. In general, an object can be a conception in a human mind, data represented in some carrier, or physical material (cf. the semiotic realms). We use *object* as the generic term to signify any concept in the object domain.

Based on the nature of the objects we can distinguish between material objects and informational objects. *Material objects* do not carry or present any information, whereas *informational objects* do. For us,

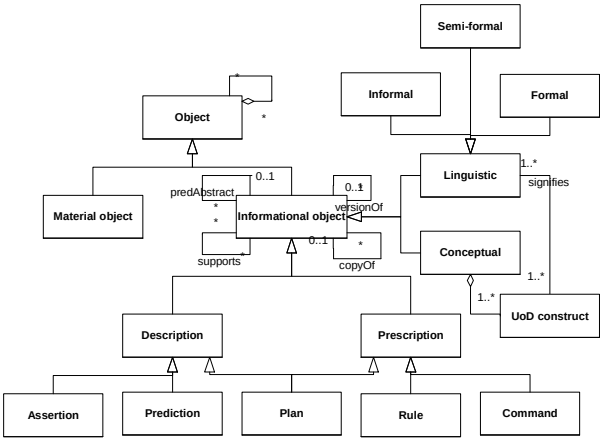


Figure 5. Object domain

objects of special interest are in the form of data or information. We call them *linguistic objects* and *conceptual objects*, respectively. Linguistic objects can be *formal*, *semi-formal* or *informal.*

Informational objects can be classified based on the intentions by which they are provided and used (e.g. [36], [32], [21]). Informational objects can be descriptive or prescriptive. A descriptive object, called a *description*, is a representation of information about a slice of reality. An informational object can be descriptive in various ways. An *assertion* is a description, which asserts that a certain state has existed or exists, or a certain event has occurred or occurs. A *prediction* is a description of a future possible world with the assertion that the course of events in the actual world will eventually lead to this state (cf. [21]). A prescriptive object, called a *prescription*, is a representation of the established practice or an authoritative regulation for action. It is information that says what must or ought to be done. A prescription with at least two parts ((E or C) and A) of the ECAA structure is called a *rule*. A prescription with neither an event part nor a condition part is called a *command*. A *plan* is a description about what is intended. It can also be regarded as a kind of prediction, which is augmented with intentions of action. It is assumed that the future possible world described in the plan would not normally come out, except for the intended actions (cf. [21]).

An object is often produced gradually through several iterations. The *versionOf relationship* holds between two objects obj_1 and obj_2, if properties of, and experience from, the object obj_1 have influenced the creation of another object obj_2 intended for the same purposes (cf. [17]). We may also have several copies from an object. The *copyOf relationship* holds between two objects, the original object and a copy object,

which are exactly, or to an acceptable extent, similar. The *supports relationship* involves two informational objects, obj_1 and obj_2, such that the information "carried" by the object obj_1 is needed to produce the object obj_2. The *predAbstract relationship* between two informational objects means that one object is more abstract that the other object in terms of predicate abstraction and both of the objects signify the same thing(s) in reality. The *signifies relationship* defines the conceptual meaning of a linguistic object in terms of UoD constructs, which the object signifies. The *UoD construct* means any conceptual construct. The *partOf relationship* means that an object is composed of two or more other objects.

3.5 Facility Domain

The *facility domain* contains all those concepts and constructs that refer to the means by which something can be accomplished, i.e. something, which makes an action possible, more efficient or effective (Figure 6). We distinguish between two kinds of *facilities*, tools and resources.

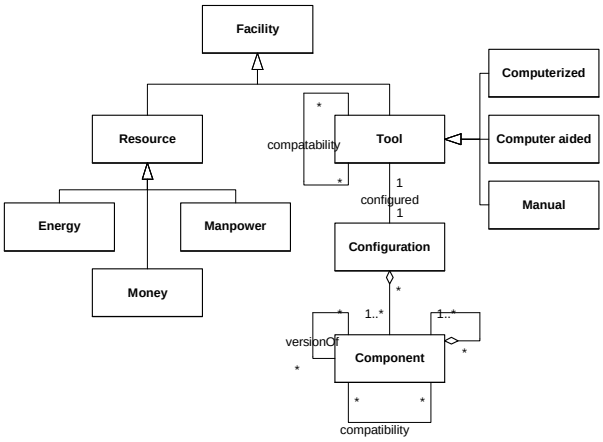


Figure 6. Facility domain

A *tool* is a thing that is designed, built, installed, etc. to serve in a specific action affording a convenience, efficiency or effectiveness. A tool may be a simple and concrete instrument held in hand and used for cutting or hitting. Or, it may be a highly complicated computer system supporting an engineer in his/her controlling a nuclear power station. Tools can be *manual, computer aided,* or *computerized.* A *resource* is a kind of source of supply, support, or aid. It can be money, energy, capital, goods, manpower, etc. [1]. The resources are not interesting in terms of pieces, but rather in terms of amount. When a resource is used, it is consumed, and when consuming, the amount of the resource diminishes. Thus, a resource is a thing, about which the main concern is how much it is available (cf. [24].

There are a great number of relationships between the concepts within the facility domain, representing e.g. functional and structural connections. We consider only some of them. For being operative and useful, tools should be compatible**.** Two tools are *compatible* if their interfaces are structurally and functionally interoperable. Tools are composed of one or more components that develop through consecutive *versions*. Only some versions of a component are compatible with certain versions of the other components. A *configuration* is a whole that is composed of the components of compatible versions.

3.6. Inter-Domain Relationships

Until now we have defined only those contextual relationships which associate concepts within the same contextual domain. There is, however, a large set of contextual relationships that relate concepts in different domains. For example, an actor carries out an action, an object is an input to an action, and a facility is situated in a location. We call these *inter-domain relationships.* Figure 7 presents an overview of inter-domain relationships. The space is divided into seven sub-areas corresponding to the seven contextual domains. In each of the sub-areas we present the concerned generic concepts to be related with the inter-domain relationships. It goes beyond the space available to define the relationships here.

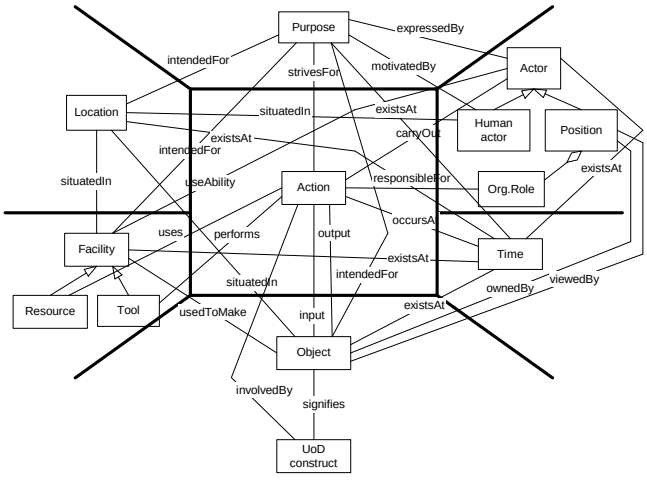


Figure 7. Overview of inter-domain relationships

In addition to the binary inter-domain relationships, there are multiple n-ary relationships. With these, together with composing binary inter-domain relationships, it is possible to specify things in the

enterprise in a way that reveals their contextual meanings. An example of this kind of specification is: the customer c places the order o for the product p at time t, based on the offer o from the enterprise e, owned by the partners $\{p_1,...p_n\}$, to be delivered by a truck tr to the address a by the date d. It depends on the situation at hand which contextual domains and concepts are seen to be relevant to be included in the specification.

4. Summary and Conclusions

In this article we have presented the context-based enterprise ontology to promote the understanding of the nature, purposes, and meanings of things about which information is stored and processed in, and transmitted between, various applications in enterprises. This ontology, grounded upon theories, such as case grammar, pragmatics and activity theory, guides a conceptualization of the structure and behavior of the enterprise through considering things as contexts, and/or as parts thereof.

Although our ontology, as having been derived from relevant theories, inherently embodies essential contextual concepts, it is just a top ontology. At this stage, it can be deployed as a frame to analyze and compare other enterprise ontologies in terms of their contextuality. Later, our ontology should be specialized into a task ontology, or a domain ontology, for the needs of a specific business task or field. Experiments made on such kinds of specializations and comparisons of their outcomes with current enterprise ontologies indicate that existing enterprise ontologies lack many essential contextual concepts and constructs and some of the conceptual constructs in them should be reengineered, in order to enable the recognition, representation and derivation of meanings in enterprise knowledge. Unfortunately, it goes beyond the space available to consider this further here. Continuing our top down approach to ontology engineering, we will next focus on a more systematic derivation of specialized concepts and constructs, and use them in empirical studies on semantic and pragmatic interoperability of enterprise applications. In this phase, we aim also to validate our ontology.

References:

[1] O. Barros, "Modeling and evaluation of alternatives in information systems", *Information Systems*, Vol. 16, No. 5, 1991, pp. 537-558.

[2] D. Bianchini, V. De Antonellis and M. Melchiori, "Ontology-based semantic infrastructure for service interoperability for interorganizational applications", In M. Missikoff (Ed.) *Proc. of the Open InterOp Workshop on Enterprise Modelling and Ontologies for Interoperability*, 7-8 June 2004, Riga, Latvia, 2004.

[3] Bratman M., *Intentions, plans, and practical reason*, Harward University Press, Cambridge, 1987.

[4] Cleland D. and W. King, *Management: a systems approach*, McGraw-Hill, New York, 1972.

[5] S. Cranefield and M. Purvis, "UML as an ontology modeling language", In Proc. of the Workshop on *Intelligent Information Integration*, held in conjunction with the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI-99), 1999.

[6] Engeström Y., *Learning by expanding: an activity theoretical approach to developmental research*, Orienta-Konsultit, Helsinki, 1987.

[7] C. Fillmore, "The case for case", In E. Bach and R. T. Harms (Eds.) *Universals in Linguistic Theory*, Holt, Rinehart and Winston, New York, 1968, pp.1-88.

[8] M. Fox, "The TOVE Project: A common-sense model of the enterprise", In F. Belli and F. Radermacher (Eds.) *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, LNAI 604, Springer-Verlag, Berlin 1992, pp. 25-34.

[9] G. Geert and W. McCarthy, "The ontological foundations of REA enterprise information systems", 2000, online: htt://www.msu.edu/user/mccarh4/rea-ontology/

[10] T. Gruber, "A translation approach to portable ontology specification", *Knowledge Acquisition*, Vol. 5, No. 2, 1993, pp. 119-220.

[11] N. Guarino, "Formal ontology and information systems", In N. Guarino (Ed.) Proc. of Conf. on *Formal Ontology in Information Systems (FOIS'98)*, IOS Press, Amsterdam, 1998, pp. 3-15.

[12] H. Herbst, "A meta-model for business rules in systems analysis", In J. Iivari, K. Lyytinen & M. Rossi (Eds.) *Advanced Information Systems Engineering*, LNCS 932, Springer, Berlin, 1995, pp. 186-199.

[13] Hoc J.-M., *Cognitive psychology of planning*, Academic Press, London, 1988.

[14] J. Iivari, "Levels of abstraction as a conceptual framework for an information system", In E. Falkenberg and P. Lindgren (Eds.) *Information System Concepts: An In-Depth Analysis*, North–Holland, Amsterdam, 1989, pp. 323-352.

[15] Jayaratna N., *Understanding and evaluating methodologies: NIMSAD – a systemic framework*, McGraw-Hill, London, 1994.

[216] Johnson G., K. Scholes and R.W. Sexty, *Exploring strategic management*, Prentice-Hall, Englewood Cliffs, 1989.

[17] R. Katz, "Toward a unified framework for version modeling in engineering databases", *ACM Surveys*, Vol. 22, No. 4, 1990, pp. 375-408.

[18] V. Kavakli and P. Loucopoulos, "Goal-driven business process analysis application in electricity deregulation", *Information Systems*, Vol. 24, No. 3, 1999, pp. 187-207.

[19] M. Koubarakis and D. Plexousakis, "A formal model for business process modeling and design", In B. Wangler and L. Bergman (Eds.) Proc. of 12th Int. Conf. on *Advanced Information Systems Engineering (CAiSE 2000)*, Springer-Verlag, Berlin, 2000, pp. 142-156.

[20] J. Lee, N.-L. Xue and J.-Y. Kuo, "Structuring requirement specifications with goals", *Information and Software Technology*, Vol. 43, No. 2, 2001, pp. 121-135.

[21] R. Lee, "Epistemological aspects of knowledge-based decision support systems", In H. Sol (Ed.) Proc. of Int. Conf. on *Processes and Tools for Decision Support Systems*, North-Holland, Amsterdam, 1983, pp. 25-36.

[22] Levinson S., *Pragmatics*, Cambridge University Press, London,1983.

[23] C.-Y. Lin, C.-Y. and C.-S. Ho, "Generating domain-specific methodical knowledge for requirements analysis based on methodology ontology", *Information Sciences*, Vol. 14, No. 1-4, 1999, pp. 127-164.

[24] L. Liu and E. Yu, "Designing web-based systems in social context: a goal and scenario based approach", In A. Banks Pidduck, J. Mylopoulos, C. Woo and M. Tamer Ozsu (Eds.) Proc. of 14th Int. Conf. on *Advanced Information Systems Engineering (CAiSE'2002)*, LNCS 2348, Springer-Verlag, Berlin, 2002, pp. 37-51.

[25] P. Loucopoulos, V. Kavakli, N. Prekas, C. Rolland, G. Grosz and S. Nurcan,. Using the EKD approach: the modelling component. ELEKTRA – Project No. 22927, ESPRIT Programme 7.1, 1998.

[26] Mesarovic M., D. Macko, and Y. Takahara, *Theory of hierarchical, multilevel, systems*, Academic Press, New York, 1970.

[27] H. Myrhaug, "Towards life-long and personal context spaces", In Proc. of Workshop on *User Modelling for Context-Aware Applications*, 2001.

[28] NATURE Team, "Defining visions in context: models, processes and tools for requirements engineering", *Information Systems*, Vol. 21, No. 6, 1996, pp. 515-547.

[29] L. Padgham and G. Taylor, "A system for modeling agents having emotion and personality", In L. Cavedon, A. Rao & W. Wobcke (Eds.) *Intelligent Agent Systems*, LNAI 1209, Springer-Verlag, Berlin, 1997, pp. 59-71.

[30] R. Ramesh and A. Whinston, "Claims, arguments, and decisions: formalism for representation, gaming, and coordination", *Information Systems Research*, Vol. 5, No. 3, 1994, pp. 294-325.

[31] C. Rolland, C. Souveyet and M. Moreno, "An approach for defining ways-of-working", *Information Systems*, Vol. 20, No. 4, 1995, pp. 337-359.

[32] Searle J. and D. Vanderveken, *Foundations of illocutionary logic*, Cambridge University Press, New York, 1985.

[33] Simon H., *The new science of management decisions*, Harper & Row, New York, 1960.

[34] Sisk H., *Management and organization*, South Western Pub. Co., International Business and Management Series, Cincinnati, 1973.

[35] J. Sowa and J. Zachman, "Extending and formalizing the framework for information system architecture", *IBM Systems Journal*, Vol. 31, No. 3, 1992, pp. 590-616.

[36] R. Stamper, "Information science for systems analysis", In E. Mumford and H. Sackman (Eds.) *Human Choice and Computers*, North-Holland, Amsterdam, 1975, pp. 107-120.

[37] R. Thayer, "Software engineering project management – a top-down view", In R. Thayer (Ed.) *Tutorial: Software Engineering Project Management*, IEEE Computer Society Press, 1987, pp. 15-56.

[38] M. Uschold, M. King, S. Moralee and Y. Zorgios, "The Enterprise Ontology", *The Knowledge Engineer Review*, Vol. 13, No. 1, 1998, pp. 31-89.

[39] X. Wang and C. Chan, "Ontology modeling using UML", In Y. Wang, S. Patel and R. Johnston (Eds.) Proc. of the 7th Int. Conf. on *Object-Oriented Information Systems (OOIS'2001)*, Springer-Verlag, Berlin, 2001, pp. 59-70.

[40] Webster, *Webster's Encyclopedic Unabridged Dictionary of the English Language*, Gramercy Books, New York, 1989.

[41] Weick K.E., *Sensemaking in organizations*, Sage Publications, California, 1995.

Establishing a Common Vocabulary for Helping Software Organizations to Understand Software Processes

Ricardo de Almeida Falbo, Gleidson Bertollo
Computer Science Department, Federal University of Espírito Santo, Vitória – ES, Brazil
falbo@inf.ufes.br, gleidsonbertollo@yahoo.com.br

Abstract

Nowadays, several process quality models and standards, such as ISO/IEC 12207 and CMMI, are used to guide software organizations in their software process improvement efforts. But unfortunately, the vocabulary used by those models and by software organizations is diverse. This leads to misunderstanding and problems related to the jointly use of different process quality models. In this paper, we present a software process ontology, which aims to establish a common vocabulary to software organizations talk about software processes. A mapping between the concepts presented in the ontology and the concepts of some of these standards is also done in order to help software organizations to use those standards in their software process improvement efforts.

1. Introduction

Developing quality software is a challenge to software organizations. Since the quality of a software product depends heavily on the quality of the software process used to develop it, software organizations are more and more investing in improving their software processes. In this context, several process quality standards, methodologies, and maturity models, such as ISO/IEC 12207 [1], ISO/IEC 15504 [2], RUP [3] and CMMI [4], are used to guide software organizations efforts towards quality software processes.

But unfortunately, the vocabulary used by those models and by software organizations is diverse. This leads to misunderstanding and problems related to the jointly use of different standards. To deal with these problems, we developed a software process ontology that is presented in this paper. This ontology aims to establish a common vocabulary to software organizations talk about software processes, and was developed as an extension of the software process ontology presented in [5]. It can be used as an interlingua to map concepts from different models and standards, helping software organizations to use them jointly. To show how this can be done, an initial mapping between the software process ontology and the concepts used in ISO/IEC 12207, ISO/IEC 15504, CMMI and RUP is also presented.

This paper is organized as follows: Section 2 discusses briefly software processes and ontologies. Section 3 presents the software process ontology developed. Section 4 presents a mapping between the ontology and the concepts of some process quality standards. Section 5 discusses related works, and, finally, in section 6, we report our conclusions.

2. Software Process and Ontologies

According to Fuggetta [6], a software process can be defined as a coherent set of policies, organizational structures, technologies, procedures, and artifacts that are needed to conceive, develop, deploy, and maintain a software product. A process should be defined considering: the activities to be accomplished, the required resources, the input and output artifacts, the adopted procedures (methods, techniques, templates and so on) and the life cycle model to be used.

To be effective and to lead to good quality products, a software process should be adequate to the application domain and to the specific project itself. Thus, processes should be defined considering several features, such as the type of software being developed, the paradigm adopted, the application domain, team features, and so on.

But, although different projects require processes with specific features, it is possible to establish a set of

software process assets that should be present in all project processes. This set of process assets is called an organization's standard software process. Thus, an organizational standard process encompasses the essential process assets (activities, artifacts, resources, procedures) that should be incorporated to all software processes of the organization. Ideally, this process should be defined considering international standards, such as CMMI and ISO/IEC 12207.

This approach can be extended to deal with several levels of standard processes. That is, the organizational standard software process can be specialized to consider some class of software type (such as information system), paradigms (for example, object-oriented paradigm) or specific application domains, giving rise to standard specialized processes.

During process specialization, process assets can be added or modified, according to the context of the specialization (software type, paradigm or application domain). Process specialization can be done recursively. For example, the organizational standard process can be specialized to derive a standard process for object-oriented development, which, in turn, can be specialized for developing object-oriented web applications.

The project's defined software process is developed by tailoring the organization's standard software process or one of its specialized standard processes to fit the specific characteristics of the project. During process tailoring, particularities of the project and team features, among others, should be considered. At this moment, the life cycle model to be followed should be defined, and new activities, as well as consumed and produced artifacts, required resources and procedures, can be added to the project's process.

Successful organizations continuously improve their processes, and systematic process improvement is more effective and efficient if it is done guided by process quality models and standards. The purpose of most standards is to help software organizations achieve excellence by following the processes and activities adopted by the most successful organizations. But it is not easy to select suitable standards. There are many choices, with a large overlap between them. Several times, it is worthwhile for a software organization to use or implement more than one standard at the same time. In this situation, it is better to implement them simultaneously. Such an approach enables process engineers to capitalize on the commonalties between the standards and use the strengths of one standard to offset the weaknesses in the other [7]. But in this case, vocabulary problems arose. Let's take a look at some of these standards.

ISO/IEC 12207 [1] provides a comprehensive set of life cycle processes, activities and tasks for software. Its Process Reference Model provides definitions of processes described in terms of process purpose and outcomes, together with an architecture describing relationships between the processes. It sets out the activities and tasks required to implement the high level life cycle processes to achieve desirable capability for acquirers, suppliers, developers, maintainers and operators of systems containing software. Three life cycle process categories are considered: Organizational, Primary and Supporting. The process model does not represent a particular process implementation approach nor does it prescribe a life cycle model, methodology or technique. Instead the reference model is intended to be tailored by an organization based on its business needs and application domain.

CMMI [4] [7] is structured in terms of process areas (PAs), which consist of related practices that collectively satisfy a set of goals. A generic goal describes the institutionalization required to achieve a capability (continuous representation) or maturity (staged representation) level. Each generic goal is associated with a set of generic practices that describes activities required for institutionalizing processes in a particular PA. Each PA still contains specific goals and specific practices, which describe activities important to achieve the specific goals.

The Rational Unified Process (RUP) [3] is represented using four primary modeling elements: workers, activities, artifacts and workflows. A worker is a role an individual or a group of individuals plays in a project. An activity of a specific worker is a unit of work that an individual in that role may be asked to perform. Activities produce artifacts and can be broken into steps. An artifact is a piece of information that is produced, modified, or used by a process, and can be composed of other artifacts. Artifacts are used as input by workers to perform an activity and are the result or output of such activities. Finally, a workflow is a sequence of activities that produces a result of observable value. These four primary elements represent the backbone of the RUP static structure. But other elements are added to make the process easier to understand and use. These additional elements are: guidelines – rules, techniques, recommendations, or heuristics that describes how to perform an activity or a step; templates – "models" of artifacts, such as a template for the project plan; tool mentors – special guidelines showing how to perform an activity or a step using a specific software tool; and concepts that are

introduced in separate sections of the process, usually attached to a core workflow.

In ISO/IEC 15504 [2], process is defined as a set of interrelated or interacting activities which transforms inputs into outputs. Analogous to CMMI and ISO/IEC 12207, standard processes are defined as the set of definitions of the basic processes that guide all processes in an organization. These process definitions cover the fundamental process elements (and their relationships to each other) that must be incorporated into the defined processes that are implemented in projects across the organization. A tailored process is a defined process developed by tailoring a standard process definition. A work product is an artifact associated with the execution of the process.

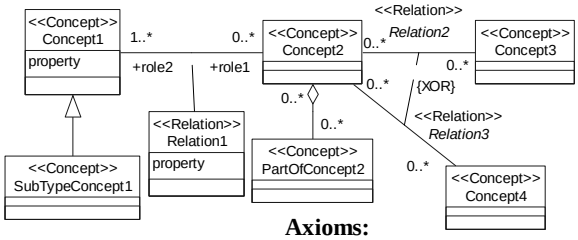
There are a large number of process standards, each one using a slightly different terminology, sometimes with different meaning for the same term, as we can see analyzing the terms and definitions of the four standards previously presented. Thus, we need to establish a common understanding of what is a software process, and which are its main assets. To achieve it, we advocate the use of ontologies.

An ontology is a representation vocabulary, often specialized to some domain or subject matter. More precisely, it is not the vocabulary as such that qualifies as an ontology, but the conceptualizations that the terms in the vocabulary are intended to capture. Ontologies are quintessentially content theories, because their main contribution is to identify specific classes of objects and relations that exist in some domain [8]. Ontologies are used to describe ontological commitments for a set of agents (humans and software applications), that is, agreements to use a shared vocabulary in a coherent manner, so that they can communicate about a domain of discourse.

An ontology, as an engineering artifact, is constituted by a vocabulary used to describe a certain reality, plus a set of explicit assumptions (formal axioms) regarding the intended meaning of the vocabulary words. This set of assumptions has usually the form of a first-order logical theory, where vocabulary words appear as unary or binary predicate names, respectively called concepts and relations [9].

As any software engineering artifact, ontologies must be developed following software engineering practices. To build the software process ontology, we used SABiO (Systematic Approach for Building Ontologies) that encompasses the following activities [5, 10]: purpose identification and requirement specification, ontology capture, ontology formalization, integration of existing ontologies, ontology evaluation, and documentation.

In the requirement specification phase, SABiO uses competency questions to establish the competence of the ontology. During ontology capture, a graphical language for expressing ontologies is used to facilitate the communication between ontology engineers and experts. In its current version, SABiO proposes the use of an UML profile for ontologies [11]. This UML profile uses some UML's model elements playing the same role of the elements of LINGO, the original language proposed [10]. I.e., these UML's model elements are applied using the same semantics imposed by the corresponding elements in LINGO, for which there were some axioms defined. For instance, the axioms (AE1) to (AE4) in Figure 1 are imposed by the whole-part relation, and are assumed to be incorporated to the ontology whenever the aggregation notation of UML is used. Figure 1 shows a summary of the UML profile for expressing ontologies and some of the axioms imposed for the corresponding notation. When any of these notations are used, the corresponding axioms (said epistemological axioms) are supposed to be incorporated, and then they do not need to be written down.



Axioms:

Whole-part:
(AE1) $\forall x \ \neg partOf(x,x)$
(AE2) $\forall x,y \ partOf(y,x) \leftrightarrow wholeOf(x,y)$
(AE3) $\forall x,y \ partOf(y,x) \rightarrow \neg \ partOf(x,y)$
(AE4) $\forall x,y,z \ partOf(z,y) \wedge partOf(y,x) \rightarrow partOf(z,x)$
Sub-type-of:
(AE5) $(\forall x,y,z)\ (subTypeOf(x,y) \wedge subTypeOf(y,z) \rightarrow subTypeOf(x,z))$
(AE6) $(\forall x,y)\ (subTypeOf(x,y) \rightarrow superTypeOf(y,x))$
Or-exclusive (XOR):
(AE7) $(\forall a \in C2)\ ((\exists b)\ (b \in C3) \wedge R2(a,b)) \rightarrow \neg((\exists c \in C4) \wedge R3(a,c)))$
(AE8) $(\forall a \in C2)((\exists c)\ (c \in C4) \wedge R3(a,c)) \rightarrow \neg((\exists b \in C3) \wedge R2(a,b)))$

Figure 1. UML's Profile and its Axioms.

A graphical model, even associated to epistemological axioms, is useful, but it is not enough to completely capture an ontology. Other axioms, called ontological axioms [10], should be provided in order to fix the semantics of the terms, and to establish domain constraints. For formalizing those axioms, SABiO suggests the use of first order logics.

Finally, for ontology evaluation, SABiO suggests checking the ontology against its competency questions, and to verify some quality criteria, as those proposed by Gruber [12].

3. A Software Process Ontology

Analyzing the elements involved in software processes, we can notice that it is a complex domain for building an ontology. As a basic premise, it is essential to follow an approach focusing on minimum ontological commitment. Based on that approach, the ontology should describe only general aspects, valid for any process, with only their essential assets. Including many details in an ontology can make it too specific and thus less reusable.

However, even considering the minimum ontological commitment criterion, this domain is still extremely complex. Therefore, it was necessary to apply a decomposition mechanism allowing building the ontology in parts. The adopted strategy was to define sub-domains of the software process domain, and build sub-ontologies for each sub-domain. Once defined the basic ontologies, these were used in an integrated way to establish a more complete conceptualization about software processes.

Figure 2 shows the software process ontology and its three sub-ontologies: activity ontology, resource ontology and procedure ontology.

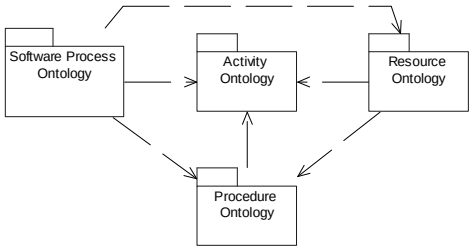


Figure 2. Software Process Ontology and its sub-ontologies.

The software process ontology was originally published in [5]. However, the software process area evolved in the last years, and we needed to evolve this ontology, capturing and defining new concepts, relations and constraints. The ontologies for software activities, procedures and resources were not modified, but only the software process ontology has changed. Thus in this paper we only present the reviewed software process ontology.

Some of the competency questions considered in this new version of the software process ontology includes:

CQ1. How can a process be decomposed?
CQ2. Which are the assets that compose a software process?
CQ3. Which are the inputs and outputs of a process?
CQ4. How can a process be classified?
CQ5. Which is the abstraction level of a process?
CQ6. How can a process be tailored?
CQ7. How do processes interact?
CQ8. How are the activities of a project's software process organized?

To treat these competency questions, some aspects should be taken into account:

- Process Decomposition and Interaction (CQ1 and CQ7);
- Process Definition (CQ2 and CQ3);
- Process Type and Abstraction Level (CQ4 to CQ6);
- Project Process Life Cycle Model (CQ8).

Following, each one of the aspects listed above are discussed and the corresponding models and axioms presented.

3.1. Process Decomposition

A process is defined to establish a systematic approach for developing software, and it can be decomposed into activities or other processes, called sub-processes. For example, according to ISO 12207, the software process can be decomposed into processes for acquisition, supply, development, operation and maintenance, among others. The development process can be further decomposed into other sub-processes, such as requirements engineering process, and so on. The requirements engineering process, in turn, can be decomposed into activities such as requirement elicitation, analysis and negotiation, modeling, documentation, evaluation, and management. Activities can also be decomposed into sub-activities, as shown in Figure 3.

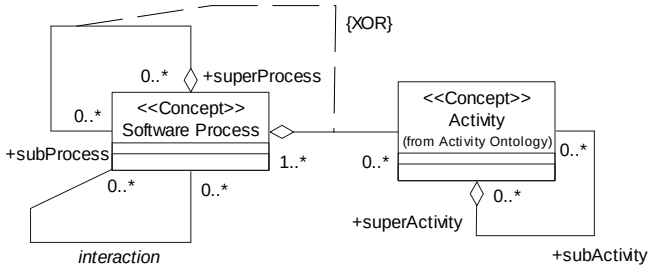


Figure 3. Process decomposition and interaction.

A super-process is the one that is composed by other processes. It cannot be executed directly through activities, as shown by the constraint {XOR} in the model. A sub-process is a software process that composes a larger process, its super-process.

Only to illustrate the epistemological axioms instantiation, the constraint {XOR} in the model of Figure 3 imposes the following axioms, derived from axioms (AE7) and (AE8) in Figure 1.

$$(\forall p1)\,((\exists p2)\ subProcess(p2,p1)) \rightarrow ((\neg\exists a)\ partOf\ (a, p1))$$

$$(\forall p1)\,((\exists a)\ partOf\ (a, p1)) \rightarrow ((\neg\exists p2)\ subProcess(p2,p1)$$

Finally, a software process can interact with other processes. This interaction can be in several ways, among them: a process can precede the execution of other, two processes can be executed in parallel, or a process can be executed in a specific moment during the execution of another process.

3.2. Process Definition

As discussed above, a process is composed by sub-processes or activities. During process definition, several other process assets should be defined. For each activity of the software process, we should define its sub-activities, pre-activities, input and output artifacts, required resources (humans, software and hardware) and the procedures (methods, techniques etc) to be followed when performing the activity. Figure 4 presents the process assets involved in software process definition.

The major part of this model corresponds to the activity ontology presented in [5]. Since in this paper we are focusing only on the evolution of the software process ontology, we will discuss only those aspects related to this review.

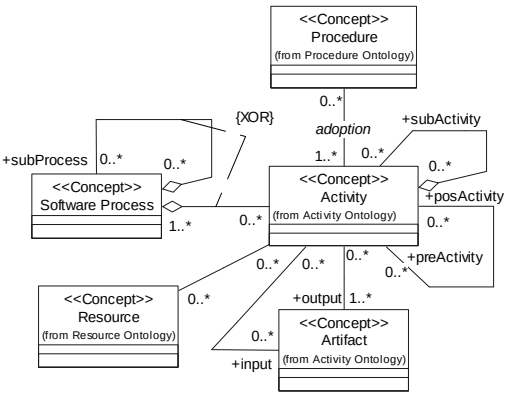


Figure 4. Process definition.

An activity is a transformational action that can produce artifacts. To be performed, an activity requires resources, adopts procedures and consumes artifacts. In a similar way, we can say that a software process has inputs and outputs. Its inputs and outputs are directly related to its activities' inputs and outputs. That is, if an activity *a1*, part of a software process *p*, requires as input an artifact *s*, and there isn't another activity *a2*, part of the same process *p*, that produces this artifact, then *s* is said an input of *p*.

$$\forall (p, a1, s)\ partOf\ (a1, p) \wedge input(s, a1) \wedge ((\neg\exists a2)\ partOf(a2, p) \wedge output\ (s, a2)) \rightarrow input\ (s, p)$$

Concerning outputs, we can say that the outputs of a process correspond to the outputs of their activities.

$$\forall (p, a1,s)\ partOf\ (a1,p) \wedge output\ (s,a1) \rightarrow output(s,p)$$

In an analogous manner, a super-process has its inputs and outputs defined through the inputs and outputs of its sub-process, as described by the following axioms:

$$\forall (p1,p2,s)\ subProcess\ (p2,p1) \wedge output\ (s, p2) \wedge ((\neg\exists p3)\ subProcess\ (p3, p1) \wedge input(s, p3)) \rightarrow input\ (s, p1)$$

$$\forall (p1,p2,s)\ (subProcess(p2,p1) \wedge output(s, p2) \rightarrow output\ (s, p1)$$

3.3. Process Type and Abstraction Level

As shown in Figure 5, processes can be classified in process categories. For example, if an organization follows the ISO/IEC 12207 classification, the categories could be primary processes, supporting processes, and organization processes. Furthermore, processes are in different levels of abstraction. A standard process refers to a generic process institutionalized in an organization, establishing basic requirements for processes to be performed in that organization. A project process refers to the process defined for a specific project, considering the particularities of that project.

Software processes (standard or project processes) can be defined tailoring a standard process. When a standard process is tailoring another standard process, the tailored process is called a specialized process, and every process assets defined in the standard process become part of the specialized process. But new assets can also be included, to deal with features of a specific software type, paradigm or application domain.

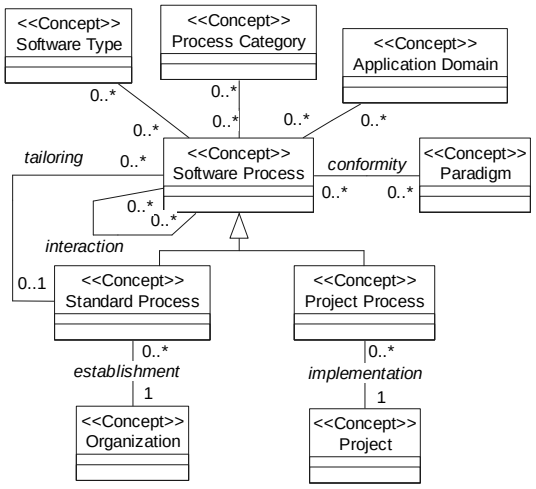


Figure 5. Process Types and Abstraction Levels.

When a project process is defined by tailoring a standard process, it is composed by all the standard process assets, and new assets can be included considering the project characteristics, such as complexity, size, and team experience, among others.

It is worthwhile to point that the interaction between processes must occur at the same level of abstraction. I.e. a standard process can interact only with other standard process and a project process can interact only with other project process.

$$\forall(p1,p2)\ (interaction\ (p1,p2) \rightarrow (standardProcess\ (p1) \wedge standardProcess\ (p2)) \vee (projectProcess\ (p1) \wedge projectProcess\ (p2))$$

3.4. Project Process Life Cycle Model

The project process definition starts with the choice of a life cycle model to be used as reference. A life cycle model structures the project activities in phases (or macro-activities), establishing an approach for organizing those macro-activities. Looking for the main life cycle models described in the literature, we can notice that macro-activities are grouped in arrangements that follow two basic strategies: sequence and iteration. In sequential arrangements, the phases are just accomplished once, returning to the previous phase only for correcting possible problems detected. In iterative arrangements, a set of phases is accomplished several times, according to some established criterion.

The waterfall life cycle model (or linear sequential model) [13], for instance, can be described as a single sequential arrangement of all phases. The spiral model [13] can be described also by only one arrangement, but in this case it is an iterative arrangement. Other life cycle models, like the recursive / parallel model [13] or the incremental model, can be described as several arrangements of activities, some of them sequential, some iterative. Thus, all life cycle models can be mapped as hybrid (sequential and iterative) arrangement of macro-activities. This way, a life cycle model defines a set of macro-activities (or phases) that a development process should present and the order of execution in the form of arrangements, as shown in figure 6.

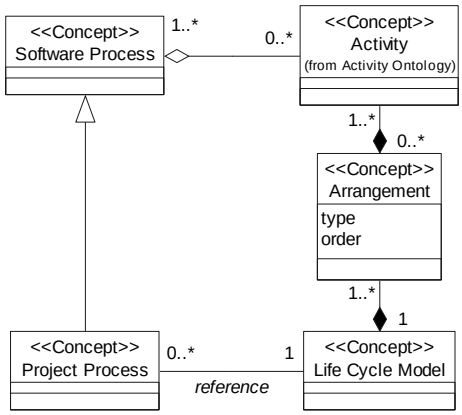


Figure 6. Project Process and Life Cycle Models.

Since the starting point for defining project processes is the life cycle model adopted, the initial project process' structure must correspond to the set of macro-activities that compose the life cycle model. That is, if a project process *p* adopts as a reference the life cycle model *m*, then each activity *a* that is part of an arrangement *c* of the life cycle model *m* must also be part of *p*.

$$\forall(p, m, c, a, n)\ (reference\ (p,m) \wedge partOf\ (c,m) \wedge partOf\ (a, c)) \rightarrow partOf\ (a, p))$$

4. Mapping Standards to the Ontology

Once defined the software process ontology, it is worthwhile to map the structure of the standards into the concepts of the ontology. It is worthwhile to point that some standards, such as ISO 9001:2000 and CMMI, are not software specific. But our mapping is focusing only on software organizations and, thus, we looked for those standards using only this perspective.

Table 1 shows a preliminary mapping between the vocabulary used by ISO/IEC 12207, ISO 9001:2000 and ISO/IEC 15504 and the concepts of the ontology presented in this paper.

Table 1. ISO x Software Process Ontology

ISO	Software Process Ontology
Process	Software Process
Standard Process	Standard Process
Tailored Process	Project Process
Work Product	Artifact
Activity / Task	Activity
Process Category	Process Category
Process	Software Process
Life Cycle Model	Life Cycle Model

Table 2 shows a preliminary mapping between the vocabulary used by CMMI and the concepts of the ontology presented in this paper.

Table 2. CMMI x Software Process Ontology

CMMI	Software Process Ontology
Process	Software Process
Standard Process	Standard Process
Defined Process (or Project's Defined Process)	Project Process
Work Product	Artifact
Practice	Activity
Process Area	Software Process
Project	Project
Life Cycle Model	Life Cycle Model

Finally, Table 3 shows a preliminary mapping between the vocabulary used by RUP and the concepts of the software process ontology.

Table 3. RUP x Software Process Ontology

RUP	Software Process Ontology
Process / Workflow	Software Process
Process Framework	Standard Process
Worker	Human Resource
Artifact	Artifact
Activity / Step	Activity
Guideline / Tool Mentor / Template	Procedure

5. Related Work

There are several works exploring the mapping between standards. Mustafelija and Stromberg [7], for example, maps ISO 9001:2000 sections to CMMI and vice versa. These mappings, however, are based on the content of the standards, and not on their structures. In fact, there are very few works dealing with the problem of establishing a common understanding about software processes. The most important of them is the OMG's Software Process Engineering Metamodel (SPEM) [14], which is used to describe a concrete software development process or a family of related software development processes.

Like our ontological approach, SPEM intends to define the minimal set of process modeling elements necessary to describe any software development process, without adding specific models or constraints for any specific area.

At the core of SPEM is the idea that a software development process is a collaboration between abstract active entities called *process roles* that perform operations called *activities* on concrete, tangible entities called *work products*.

SPEM follows an object-oriented approach for modeling a family of related software processes, and its specification is structured as a UML profile, and provides a complete MOF-based metamodel.

In our point of view, the main problem of SPEM is exactly this approach. Several non intuitive concepts are used to define concepts related to software process. Abstract concepts, such as Model Element, Package, Work Definition and Process Performer, are not intuitive for process engineers. In fact, they are used only because an object-oriented approach, focused on inheritance, is applied. If we look for the concrete classes in SPEM, we can find a great correspondence with our ontology, as we can notice in Table 4. This table shows the mapping of some concepts of SPEM into concepts of the Software Process Ontology.

Table 4. SPEM x Software Process Ontology

SPEM	Software Process Ontology
Process Role	Human Resource / Team
Work Product	Artifact
Activity / Step	Activity
Guidance	Procedure
Categorizes Dependency	Process Category
Process	Software Process
Life Cycle	Life Cycle Model

It is worthwhile to point that, although we use an UML profile as a modeling language for expressing ontologies, we do not follow an approach like SPEM. In our case, we defined an UML profile using stereotypes to capture our meta-ontology, which includes concepts such as Concept, Relation, Property and so on [11]. We are not using UML's meta-model as basis for defining our software process ontology, as SPEM does.

6. Conclusions and Future Work

Nowadays, software process improvement is being considered essential to software organizations survive in a competitive market. But systematic process improvement is achieved only if it is done guided by process quality models and standards. Several times, it is important to use more than one standard, so that the strengths of one standard can be used to offset the weaknesses in the other, and vice versa. But in this case, we face problems related to the vocabularies used by the different standards. Generally, each standard uses its own terminology, adopting different terms to designate the same meaning. To overcome this problem, we need to establish a common conceptualization about software processes, and thus ontologies can be useful. Thus, in this paper, we presented an ontology of software process that defines the main concepts, relations, properties and constraints involved in this complex domain. Also, a preliminary mapping between the concepts in the ontology and the concepts used by some of the most important standards was done. We hope that this mapping can be used by software organizations to better understand the commonalties and differences between the various standards.

As future work, we are planning to do a more complete mapping between these standards, using our ontology as an interlingua. A mapping considering the contents of those standards is also useful.

Finally, we are working on a process infrastructure for ODE [15], a Process-Centered Software Engineering Environment that is developed based on our ontology and that can be configurable for using the most adequate vocabulary, given by the choice of a standard that a software organization commits to. ODE is developed following a systematic approach for deriving object models from ontologies, and the ontology presented in this paper is used to derive the core classes of process control in ODE, supporting tool integration and interoperability in it.

7. Acknowledgments

This work was accomplished with the support of CNPq, an entity of the Brazilian Government reverted to scientific and technological development.

8. References

[1] ISO/IEC 12207:1995, Amd 1:2002, Amd 2:2004, *Information Technology - Software life cycle processes*.

[2] ISO/IEC 15504:2003, *Information Technology – Process Assessment.*

[3] P. Kruchten, *The Rational Unified Process: An Introduction*, Addison Wesley, 1998.

[4] M.B. Chrissis, M. Konrad, S. Shrum, *CMMI: Guidelines for Process Integration and Product Improvement*, Addison Wesley, 2003.

[5] Falbo, R.A., Menezes, C.S., Rocha, A.R.R. A Systematic Approach for Building Ontologies. *Proceedings of the 6th Ibero-American Conference on Artificial Intelligence*, Lisbon, Portugal, Lecture Notes in Computer Science, vol. 1484, 1998.

[6] A. Fuggetta, "Software Process: A Roadmap", In *Proceedings of The Future of Software Engineering (ICSE'2000)*. Limerick, Ireland, 2000, 25-34.

[7] B. Mutafelija, H. Stromberg, *Systematic Process Improvement Using ISO 9001:2000 and CMMI*, Artech House, 2003.

[8] Chandrasekaran, B., Josephson, J.R., Benjamins, V.R. What Are Ontologies, and Why Do We Need Them? *IEEE Intelligent Systems*, January/February 1999.

[9] Guarino, N. Formal Ontology and Information Systems. In: *Formal Ontologies in Information Systems*, N. Guarino (Ed.), IOS Press, 1998.

[10] Falbo, R. A. Experiences in Using a Method for Building Domain Ontologies. In: *Proc. of the 16th International Conference on Software Engineering and Knowledge Engineering, International Workshop on Ontology In Action*. Banff, Canada, 2004.

[11] Mian, P.G., Falbo, R.A. Supporting Ontology Development with ODEd, *Journal of the Brazilian Computer Science*, vol. 9, no. 2, November 2003, 57-76.

[12] Gruber, T. Towards principles for the design of ontologies used for knowledge sharing, *International Journal of Human-Computer Studies*, 43(5/6), 1995.

[13] R.S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw Hill, 2004.

[14] OMG, *Software Process Engineering Metamodel Specification*, Version 1.1, January 2005.

[15] R.A. Falbo, F. B. Ruy, R. Dal Moro. Using Ontologies to Add Semantics to a Software Engineering Environment. In: *Proc. of the 17th International Conference on Software Engineering and Knowledge Engineering*. Taipei, China, 2005.

Mapping Versus Transformation in MDA: Generating Transformation Definition from Mapping Specification

Slimane Hammoudi, Jérôme Janvier, Denivaldo Lopes
ESEO, Ecole Supérieure d'Electronique de l'Ouest
4, Rue Merlet de la Boulaye - BP 926, 49009 ANGERS cedex 01
{shammoudi, janvieje, dlopes}@eseo.fr

Abstract

This paper aims firstly to clarify between the two concepts of mapping and transformation in Model Driven Architecture (MDA) where a real lack of consensus exists on their definitions. For this clarification, we have been inspired from the two fields of databases and ontologies where the two concepts have been studied for a long time. Secondly, this paper aims to propose a new approach and architecture for the process of transformation in MDA, in which the transformation definition is generated automatically from a mapping specification. Thus, in our approach the transformation process of a Platform Independent Model (PIM) into a Platform Specific Model (PSM) can be structured in two stages: mapping specification and transformation definition. From a conceptual point of view, the explicit distinction between mapping specification and transformation definition remains in agreement with the MDA philosophy, i.e. the separation of concerns. Moreover, a mapping specification could be associated with different transformation definitions, where each transformation definition is based on a giving transformation definition metamodel (language).

1 Introduction

The main motivation behind Model Driven Architecture (MDA) [1] is to transfer the focus of work from programming to modeling by treating models as the primary artifacts of development. MDA has a potential to increase development productivity and quality by describing important aspects of a solution with more human-friendly abstractions and by generating common application fragments with templates. The most important aspect of the MDA approach is the explicit identification of Platform Independent Models (PIMs) and the flexibility to implement them on different platforms via Platform Specific Models (PSMs). A platform can be any technology that supports the execution of these models, either directly or after translation to code. For this vision to become reality, software development tools need to automate the many tasks of model construction and transformation. Thus, since the emergence of MDA, numerous techniques have been proposed for transforming models at different levels of abstractions. However, most of these works state in an obvious manner the lack of consensus on the definition of the two main concepts of mapping and transformation involved into the whole process of transformation in MDA. In this paper, we present in the first part a clarification of the concepts of mapping and transformation in the context of MDA, inspiring by two main fields: Database systems and Ontologies, where these two concepts have been studied for long time. Thanks to this clarification, we propose in a second part a new architecture of a transformation system based on the four levels metamodeling architecture of MDA. In this new architecture, mapping and transformation are explicitly distinguished and together involved in the whole process of transformation in MDA. Mappings are considered as first class entities defined by a model, which conforms to a metamodel of mapping. Transformation definition (transformation model) is generated automatically from a mapping model in our approach and is executed by a transformation engine, which takes a source model and produces a target model. A transformation definition is based on a transformation metamodel, which is an abstract definition of a transformation language such as ATL [8] used in our different experiments.

This paper is structured as follows: section 2 introduces the MDA approach and presents the most common scenario of transformation in MDA, which is compatible with MOF/QVT RFP [2]. Section 3 shows the lack of consensus around the concepts of mapping and transformation. Section 4 clarify the two concepts inspired from database and ontology's fields and introduce our approach and architecture for the whole

process of transformation in MDA. Finally, section 5 concludes our work and presents some final remarks.

2 MDA: Overview and Transformation process

At the beginning of this century, software engineering needs to handle software systems that become larger and more complex than before. The object-oriented and component technology seem insufficient to provide satisfactory solutions to support the development and maintenance of these systems. To adapt to this new context, software engineering has applied an old paradigm, i.e. models, but with a new approach, i.e. Model Driven Architecture.

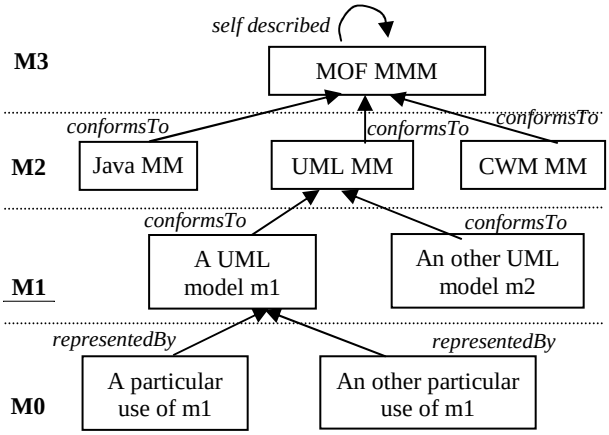


Figure 1a. Architecture with four Meta-layers

In level M3, a metametamodel is a well-formed specification for creating metamodels such as the Meta Object Facility (MOF), a standard from OMG. In level M2, a metamodel is a well-formed specification for creating models. In level M1, a model is a well-formed specification for creating software artifacts. In level M0, an operational example of a model is the final representation of a software system. According to this architecture, we can state the existence of few metametamodels such as MOF [3] and Ecore [4], several metamodels such as UML, UEML [5] and EDOC [6], more models describing real life applications such as a travel agency, and finally infinite information such as the implementation of this travel agency model using Java or C#. This organization is well known in programming languages where a self-representation of EBNF notation could be obtained easily in some lines. This notation allows defining infinity of well-formed grammars. A given grammar, e.g. the grammar of the C language, allows defining the infinity of syntactically correct C program. Several different executions could be realized from a C program. We would like to point out here a very important remark concerning the two relationships *"representedBy"* and *"conformsTo"*. It is very important to distinguish these new relationships from the old relations of *"instanceOf"* and *"inheritsFrom"* of object technology. As stated in [7], currently, there is an over-usage of these old relationships in Model Driven Engineering. Used in different contexts, with different meanings, this may cause additional confusion for example by stating that a model is an *instanceOf* a metamodel. It is very important to make a careful distinction between concepts behind the old principle of object technology *"Everything is an object"* and the concepts of the new principle of model driven engineering *"Everything is a model"*.

2.1 Model Driven Architecture (MDA)

MDA is a particular variant of a new global trend called Model Driven Engineering (MDE). MDA is based on an architecture with four meta-layers [3]: metametamodel, metamodel, model and information (i.e. an implementation of its model).

Figure 1a presents the basic metamodeling architecture of MDA with the relationships between different levels of models. In this approach, everything is a model or a model element. In level M0, a real system is *representedBy* a model in level M1, and a model in level M1 *conformsTo* a metamodel in level M2. We will discuss these two very important relationships of MDA later.

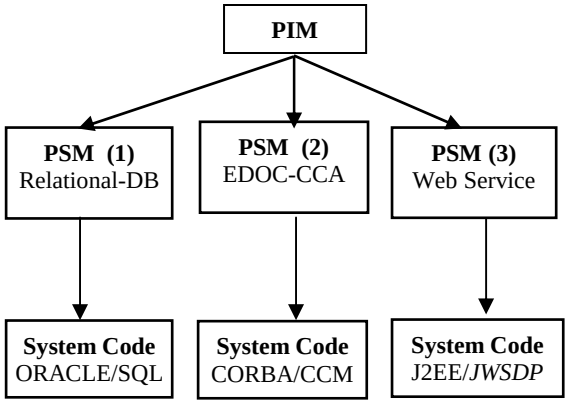


Figure 1b. MDA: Primary Idea

Figure 1.b illustrates the primary idea around the development of software systems using MDA. The development is based on the separation of concerns (e.g. business and technical concerns), which are afterwards transformed between them. So, business concerns are represented using Platform-Independent Model (PIM), and technical concerns are represented using Platform-Specific Model (PSM). According to

figure 1.b, PIM (e.g. a UML business model) is transformed into PSM (e.g. based on Web Services), which could be refined in other PSMs (e.g. based on Java and JWSDP), until exported as code, config files, and so on. Analyzing each type of model, we can deduce that a PIM and PSM have a different life cycle. PIM is more stable over time while PSM is subject to frequent modification. So, this approach preserves a business's logic (i.e. PIM) against the changes or evolution of technologies (i.e. PSM).

2.2 Model Transformation in MDA

It is well recognized today that model transformation is one of the most important operation in MDA [8]. The following definition of model transformation largely shared in the community is provided in [9]: "A Transformation is the automatic generation of a target model from a source model, according to a transformation definition.

"A transformation definition is a set of transformation rules that together describe how a model in the source language can be transformed into a model in the target language. A transformation rule is a description of how one or more constructs in the source language can be transformed into one or more constructs in the target language".

The working group on model transformation of the Dagstuhl seminar [10] suggests that this should be generalized, in that a model transformation should also be possible with multiple source models and/or multiple target models. In our discussions here we are concerned by transformations that takes a platform-independent model and transforms it in a platform-specific model.

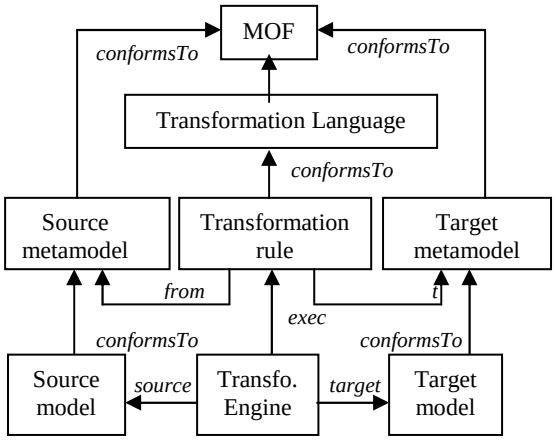


Figure 2. Model Transformation in MDA: from PIMs to PSMs

In the context of the basic four levels metamodeling architecture of MDA, various scenarios of model-to-model transformation have been identified [11]. Figure 2 presents the most common scenario of these transformations, which is compatible with MOF/QVT RFP [2].

Each element presented in Figure 2 plays an important role in MDA. In our approach, MOF is the well-established metametamodel used to create metamodels. The PIM reflects the functionalities, the structure and the behavior of a system. The PSM is more implementation-oriented and corresponds to a first binding phase of a given PIM to a given execution platform. The PSM is not the final implementation, but has enough information to generate interface files, programming language code, interface definition language, configuration files and other details of implementation. Mapping from PIM to PSM determines the equivalent elements between two metamodels. Two or more elements of different metamodels are equivalent if they are compatible and they cannot contradict each other. A transformation engine that executes transformation rules realizes model transformation. Transformation rules specify how to generate a target model (i.e. PSM) from a source model (i.e. PIM). To transform a given model into another model, the transformation rules map the source into the target metamodel. The transformation engine takes the source model, executes the transformation rules, and gives the target model as output. Using a unique formalism (MOF) to express all metamodels is very important because this allows the expression of all sorts of relationship between models based on separate metamodels. Transformations are one important example of such a relationship, but there are also others [7] like model weaving, model merging, model difference, model metrication (establishing measures on models), metamodel alignment, etc. Thus, given $m_a(s)/M_a$ and $m_b(s)/M_b$, where m_a is a model of a system s created using the metamodel M_a, and m_b is a model of the same system s created using the metamodel M_b, then a transformation can be defined as $m_a(s)/M_a \rightarrow m_b(s)/M_b$. When M_a and M_b are based on the same metametamodel (e.g. MOF), the transformation may be expressed in a transformation language such as ATL [8].

There are a number of general challenges in the definition of a language for model transformation [12]. Some of these challenges are: it must be expressive and provide complete automation, be unambiguous, and Turing complete for it to be generally applicable. The current standardization effort by OMG [2] and many industrial and academic efforts in this area will allow advancement on these challenges.

3 Mapping Versus Transformation: A lack of consensus

As noticed in the previous page, the figure 2 illustrates the most common scenario for model-to-model transformation in MDA. According to this figure we would like to point out two main remarks. The first remark concerns the "Transformation rules" component, which merge together techniques of mappings and transformations without explicit distinction between them. That is to say, the specification of correspondences between elements of two metamodels and the transformation between them are grouped in the same component at the same level. We argue here that an explicit distinction between techniques of mapping and transformation could be very helpful in the whole MDA process of transformation and we will comment more on this issue afterward.

The second remark is related to the first one and concerns the lack of consensus on terminology around MDA concepts. Actually, nowadays, MDA suffers from a lack of agreement on terminology, especially concerning the concepts of mapping and transformation. In several works, the concepts of mapping and transformation are not so clear, since these terms can refer to many different concepts. Moreover, as noticed before they are usually defined without explicit distinction between them.

The table 1 discussed in [13], illustrates in an obvious manner that the terminology related to these concepts is really immature.

In this table we have on the left part several recent publications concerning transformation process in MDA and on the top part five different concepts linked to the «Pattern of Transformation» that have been identified. These concepts are between "Transformation instance" which is a process that takes a source model to produce a target model, and a "Transformation Metamodel", which is an abstract formalism for transformation, that allows the generation of transformation definition, which is actually a "Transformation Program".

According to our vision, the concepts of mapping and transformation should be explicitly distinguished, and together could be involved in the same process that we denominate transformation process. In fact, in the transformation process, the mapping specification precedes the transformation definition. A mapping specification is a definition (the most declarative as possible) of the correspondences between metamodels (i.e. a metamodel for building a PIM and another for building a PSM). Transformation definition contains a description to transform a model into another using a concrete transformation language such as ATL. Hence, in our approach the transformation process of a PIM into a PSM can be structured in two stages: mapping specification and transformation definition.

This explicit distinction between mappings and transformations is emphasized in the two following main fields: Databases and Ontologies. In the context of databases, mapping and transformation have been studied for a long time in the domain of database design. In a recent project on model management leaded by Phil Bernstein [14], they define mapping between database schemas as follow:

	Transfo. Instance	**Transfo. Function**	**Transfo. Model**	**Transfo. Program**	**Transfo. Metamodel**
MDA Guide - OMG [20]	*Transfo*	*Mapping Instance*	*Mapping Model*		____
MDA Distilled [21]	Mapping	*Mapping rule*	*Mapping function*	____	____
MDA Explained [9]	*Transfo. Mapping*	*Mapping/Transfo. rule*	*Transfo. definition*	____	____
QVT - DSTC [22]	*"Tracking"*	*Transfo. rule*	*Transfo.*	____	*Transfo. Model*
Caplat & al. [23]	___	*Mapping*	*Model of Mapping*		____
Judson & al. [24]	*Transfo.*	*Transformation Constraint*	*Transfo.* Pattern	*Model Transformation*	____

Table 1: Equivalencies between terms according to the Transformation Pattern

"We defines a mapping to be a set of mapping elements, each of witch indicates that certain elements of schema S_1 are mapped to certain elements in S_2. Furthermore, each mapping element can have a mapping expression which specifies how the S_1 and S_2 elements are related".

In [15], they have studied mapping adaptation under evolving schemas in dynamic environments like the Web. In this work mapping is defined as follow:

"A mapping specifies how data instances of one schema correspond to data instances of another. Mappings are often specified in a declarative, data-independent way (for example, as queries or view definitions). However, they necessarily depend on schemas they relate."

Finally, the distinction between mapping and transformation is more stated in Ontologies field. In the OntoMerge Project [16] at the university of Yale, they claim the importance to distinguish between Ontology translation and ontology mapping [17]:

"It's important to distinguish ontology translation from ontology mapping, which is the process of finding correspondence (mappings) between the concepts of two ontologies. If two concepts correspond, they mean the same thing, or closely related things. The mappings should be expressed by some mapping rules, which explain how those concepts correspond. Obviously, ontology translation needs to know the mappings of two ontologies first, then it can use the mapping rules.".

4 From Mapping to Transformation in MDA

Figure 3 illustrate a simple example involving the concepts of mapping and transformation according to our point of view. In this example a fragment of UML metamodel is mapped into a fragment of relational database metamodel. The mapping part is defined here using a graphical formalism that we have introduced [18] to specify mappings between elements of two metamodels, which are MOF compliant. This graphical formalism is very useful to specify mappings in a declarative manner and at a high level of abstraction. However, it is clear that this formalism is not sufficient to express complex mappings. Thus, a textual language must sometimes be used to complete it. OCL (Object Constraint Language) have been used in several experimentations of our approach [19].

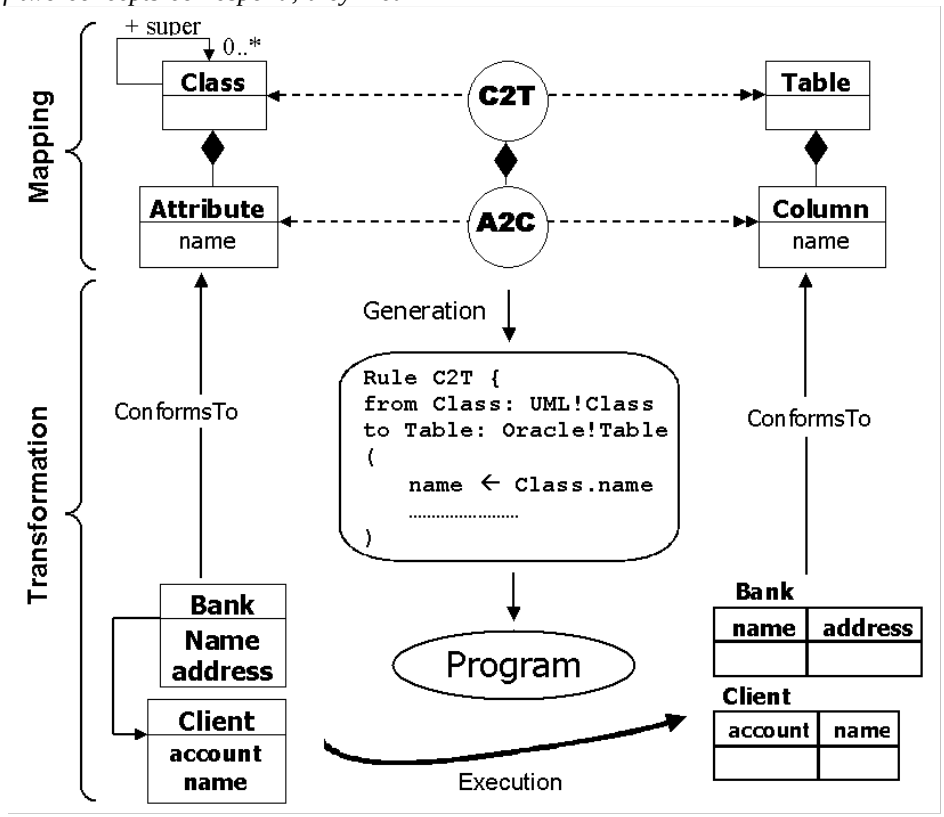


Figure 3. Transformation Process: from mapping to Transformation

All the components linked to the concepts of mapping and transformation, and their relationships, are presented in figure 4 [19] based on the four levels MDA Metamodeling Architecture, which extends figure 2 according to our approach.

This mapping part, which groups a set of correspondences between elements of two metamodels, is represented in our approach as a **mapping model**(`Mapping M`)**.** This model must conform to a **mapping metamodel** (`Mapping MM`)**.** In [19], a complete description of this metamodel is presented.

In the transformation part, a transformation definition is generated automatically from the mapping model. This transformation definition is expressed in a transformation language based on MDA standards (OCL, MOF). This transformation definition represents a **transformation model** (Transformation M), which conforms to a **transformation metamodel** (Transformation MM). This metamodel is a general formalism for model transformation in MDA. Currently, OMG is finalizing a standard for model transformation called QVT (Query / View / Transformation) [2].

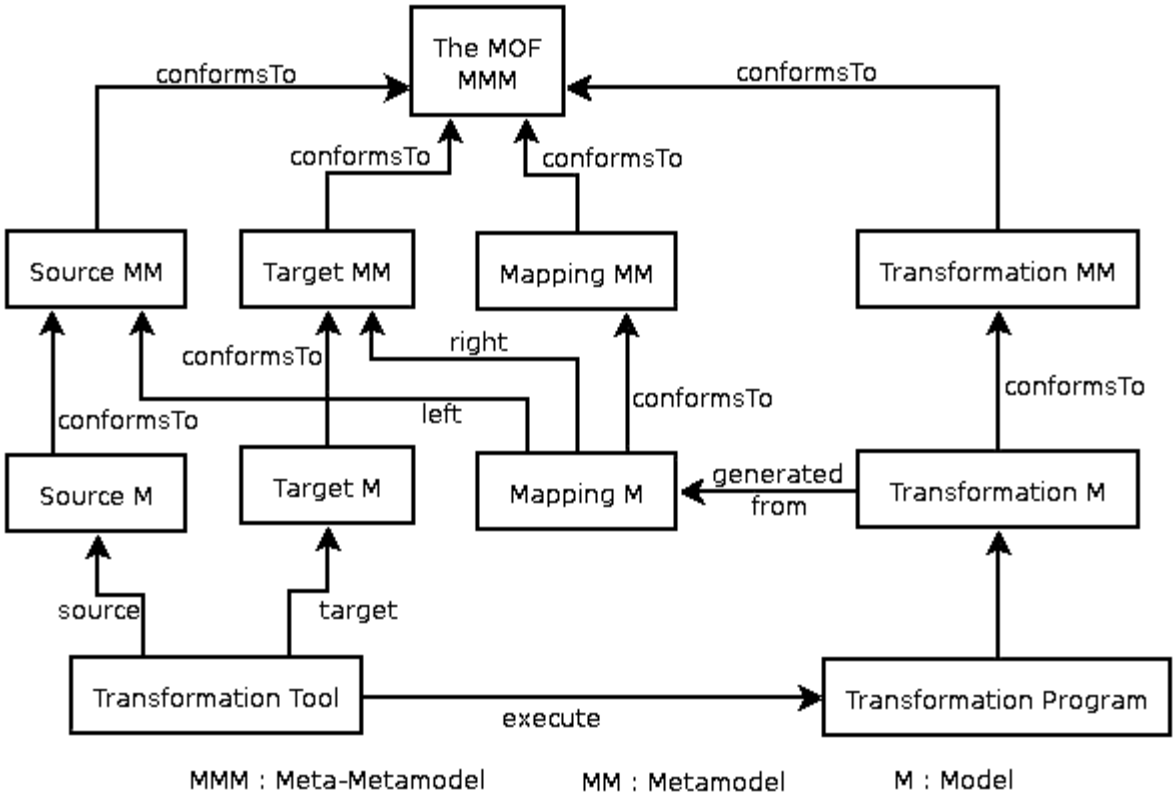


Figure 4. Mapping and Transformation in MDA

In this figure, a mapping model conforms to its metamodel, and it relates two metamodels (`Source MM` and `Target MM`). A transformation model conforms to its transformation metamodel, and it is generated from a mapping model. A transformation engine takes a source model as input, and it executes the transformation program to transform this source model into the target model.

5 Discussion and conclusion

In this paper, we have discussed the MDA approach providing a description of a transformation process, distinguishing explicitly the mapping and transformation parts. Mapping should be considered as first class entity represented by a model, which conforms to a metamodel. On the one hand, mapping part focuses on identifying elements of two given metamodels that correspond to each other. On the other hand, transformation is generated automatically as a program, which permits to translate a source model

into a target model. Mappings should be as declarative as possible, while transformations are detailed programs often involving imperative and declarative constructs. From a conceptual point of view, this explicit distinction between mapping specification and transformation definition remains in agreement with the MDA philosophy, i.e. the separation of concerns. Thus, a mapping model may be considered as a PIM that will be transformed into a PSM, the transformation model.

6 References

[1] OMG: Model Driven Architecture (MDA)-document number ormsc/2001-07-01. (2001).

[2] OMG: Request for Proposal: MOF 2.0 Query/Views/Transformations RFP. (2002).

[3] OMG: Meta Object Facility (MOF) Specification. (2002) Version 1.4.

[4] Eclipse Tools Project: Eclipse Modeling Framework (EMF) version 2.0. (2004)

[5] UEML.org: Unified Enterprise Modeling Language (UEML) (2003) Available at http://www.ueml.org.

[6] OMG: UML Profile for Enterprise Distributed Object Computing Specification. (2002)

[7] Bézivin, J. On the Unification Power of Models. Software and System Modeling (SoSym) 4(2):171--188. 2005

[8] Bézivin, J., Dupé, G., Jouault, F., Pitette, G., Rougui, J.E.: First Experiments with the ATL Model Transformation Language: Transforming XSLT into XQuery. 2nd OOPSLA Workshop on Generative Techniques in the context of Model Driven Architecture (2003)

[9] Kleppe, A.,Warmer, J., Bast,W.: MDA Explained: The Model Driven Architecture: Practice and Promise. 1st edn. Addison-Wesley (2003);

[10] Bézivin J, Heckel R, Language Engineering for Model-driven Software Development. Dagstuhl Seminar April 2004

[11] I Kurtev, K.Van den Berg. A synthesis based approach to Transformation in an MDA Software Development Process. CTIT Technical Report TR-CTIT-03-27, University of Twente June 2003.

[12] Sendall S, Kozaczynski W: Model Transformation – the Heart and Soul of Model Driven Software Development. IEEE Software, Special Issue on Model Driven Software Development, pp42 (Sept /Oct 2003).

[13] Hammoudi S, Lopes D. From Mapping Specification to Model Transformation in MDA: Conceptualization and Prototyping. First International Workshop, MDEIS'2005.

[14] Bernstein, P.A.: Applying Model Management to Classical Meta Data Problems. Proceedings of the Conference on Innovative Data Systems Research (CIDR 2003) (2003)

[15] Velegrakis Y, Miller R.J, Popa L, Mapping Adaptation under Evolving Schemas. In International Conference of Very Large Databases (VLDB), pp. 584-595, Sep. 2003.

[16] ONTOMERGE Project. http://cs-www.cs.yale.edu/homes/dvm/daml/ontology-translation.html

[17] Dou D, McDermott D, and Peishen Qi 2003 Ontology Translation on the Semantic Web. In Proc. Int'l Conf. on Ontologies, Databases and Applications of Semantics (ODBASE2003). LNCS 2888.

[18] Bézivin, J., Hammoudi, S., Lopes, D., Jouault, F.: Applying MDA Approach for Web Service Platform. 8th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2004) (2004) 58–70.

[19] Lopes D., « Étude et applications de l'approche MDA pour des plates-formes de Services Web », Thèse de doctorat, Université de Nantes, UFR Sciences et Techniques, juillet 2005.

[20] OMG, « MDA Guide Version 1.0.1 », OMG/2003-06-01, June 2003.

[21] S. J. Mellor, K. Scott, A. Uhl, and D. Weise. MDA Distilled: Principles of Model-Driven Architecture. Addison-Wesley, 1st edition, March 2004.

[22] DSTC, IBM, and CBOP. *MOF Query / Views / Transformations Second Revised Submission*, January 2004. ad/2004-01-06.

[23] G. Caplat and J. L. Sourrouille. Model Mapping in MDA. *Workshop in Software Model Engineering (WISME2002)*, 2002.

[24] S.R.Judson, R.B.France, D.L.Carver. "Specifying Model Transformation at the Metamodel Level", WISME 2003.

Integrating Enterprise Information Representation Languages

Kilian Kiko
University of Mannheim
A 5, 6, B242
D-68161 Mannheim
0049 621 181 3912
Kilian.Kiko@gmx.net

Colin Atkinson
University of Mannheim
A 5, 6, B242
D-68161 Mannheim
0049 621 181 3911
atkinson@informatik.uni-mannheim.de

Abstract

As the importance of enterprise computing systems continues to grow so does the need for sound but flexible representations of the information they manipulate. This has created a growing interest in information representation languages that are not only easy for domain and business experts to use but are also amenable to computer manipulation. Since traditional information representation languages have tended to focus either on human usability (e.g. UML) or machine processability (e.g. OWL) there is currently no language intended for human use that cleanly satisfies both requirements. In this paper we discuss the different schools of thought on how to solve this problem, and analyze the various concrete proposals that have been put forward. We then present our own views on how best to meet this challenge.

1. Introduction

Enterprise computing, in which large numbers of computing devices cooperate to achieve some common goal or deliver common services, is contingent on those devices having access to the same representation and understanding of "enterprise information". Enterprise information is any information or "knowledge" that components of an enterprise need to be aware of to be able to contribute constructively to the execution of the system, such as enterprise-spanning concepts, rules and policies.

At the realization level within a running enterprise system, XML has emerged as the standard format for representing enterprise information. Since it defines a universal concrete syntax, XML allows the individual components of an enterprise to exchange and process information in a standard way. However, XML only supports the syntactic representation of information. To attain agreement at the conceptual level the developers of components must agree on the meaning of a set of domain types via a common document type definition (DTD) or XML schema or must use a higher-level "knowledge representation language" such as RDF, RDFS, DAML or OWL to formally and explicitly define information semantics. Using these languages (which were originally defined to support the semantic web) enterprise information experts can design so called "ontologies" that represent the necessary domain knowledge explicitly. Unfortunately, however, these are also represented using an XML-based concrete syntax.

Although XML is an excellent medium for machines to share "knowledge" it is a very human unfriendly representation format. Even specialized IT personnel find it tedious and difficult to write XML, let alone normal users who are the source of most enterprise information. This makes it almost essential that the XML representation of information used within a running enterprise system be generated automatically from higher-level "human friendly" representations. These "high level" representations need to be as accessible as possible to the generators and owners of enterprise information but at the same time transformable into XML with minimal intervention by IT personnel since this is a significant source of errors. In other words, high-level Information Representation Languages (IRLs) are required which (a) are as friendly as possible to domain/business experts to allow them to capture their knowledge and (b) have as precise a meaning as possible to enable them to serve as the source of automated transformations.

One of the most widely-accepted and popular Information Representation Languages (IRL) is the UML [3]. Since it was developed for ease of use rather than semantic precision UML is more expressive than semantic web languages such as OWL but generates models which usually have more ambiguous semantics. In contrast, Ontology Representation Languages (ORL) based on Description Logic, such as OWL DL [1], have the great strength that they capture information in a way that is semantically precise and computational. Until relatively recently, therefore, there was no significant overlap in the range of applications for which the UML and ORLs were both suited, and it was easy to recommend which technology to use in which situation. UML was clearly the

best choice for software engineering oriented analysis and design where expressive power is more important than semantic precision, whereas description logic based ORLs such as OWL DL and DAML were clearly the best choice for semantically precise and computational knowledge representation.

This clear separation of concerns was significantly blurred, however, with the addition of the Object Constraint Language [6] to the UML suite of standards. OCL was defined with the specific goal of enabling the meaning of UML models to be more precisely specified. As a consequence it is a significant open question as to how the combination of the UML and OCL compares to knowledge representation languages such as OWL with regard to expressiveness, semantic precision and computability. Broadly speaking three distinct groups or schools of thought can be identified on this issue.

The first perceives UML[1] as a self-contained IRL equally as capable of representing ontologies as DL-based semantic web languages such as OWL. Proponents of this approach such as Cranefield et al. [7] and Flakovych et al. [9] hold that it makes no difference whether an ontology is modeled in OWL or in UML. UML modelers are thus instantly promoted to the status of "ontology developers".

The second group grants only lightweight ontology definition capabilities to the UML and holds that an extension to the language is required to allow it to support advanced OWL-like capabilities. Members of this group such as Baclawski [4] believe there is value in using the UML and UML tools for ontology engineering but only as a convenient "front end" for "proper" ontologies in DL based languages such as OWL. This is primarily driven by the lack of a graphical front-end for the semantic web languages. The basic philosophy of this group, therefore, is to introduce ontology representation concepts into the UML that are not inherently supported by UML diagrams. The result is a mixed language that uses UML syntax and a set of stereotyped UML constructs to emulate advanced OWL features.

The third group goes even further by introducing a new metamodel into the MDA family of modeling languages - the so called Ontology Definition Metamodel. The approach denies any knowledge representation capabilities to the UML and holds that UML syntax can only be used to represent ontologies via a new modeling language. Members of this group are not interested in using the UML per se for ontology definition but only in "borrowing" some of its user-oriented graphical syntax comparable to the second group. The metamodel is either a close copy or a direct representation of OWL's abstract syntax in the form of a MOF-based metamodel.

[1] In the remainder of this paper, the term UML will be used to refer to the combination of UML and OCL, unless stated explicitly to the contrary.

Our goal in the remainder of this paper is to explore these three schools of thought in more detail and to highlight the fundamental differences between them. Where they exist we also discuss the various unification proposals that have been put forward. In the conclusion we then state our position on the issue and present our suggestions for how the UML and ORL technology spaces can best be unified.

2. UML as an Ontology Representation Language

A number of researchers representing the first school of thought have recognized the potential of the UML for ontology modeling. Cranefield and Purvis [7], for example, investigated to what extent a subset of the UML (consisting of class diagrams and object diagrams) combined with the OCL could be used as an ontology representation language. They discovered that UML/OCL can be used for several reasoning tasks, but also came to the conclusion that the OCL is in general too expressive and needs to be restricted to a set of standard OCL constraints that are amenable to automated reasoning.

Cranefield and Purvis also came to the conclusion that a combination of UML class and object models and OCL are more expressive than description logics [7]. To be equivalent to OWL, the UML and OCL combination needs to be restricted to an adequate set of language constructs. They also claimed that it is powerful enough to be used as an ORL plus logic/rule language combination such as the enhanced semantic web languages like DAML-L or RuleML on top of OWL. Cranefield also developed a "UML Data Binding" tool for Java [8] to generate Java classes and RDF schemata from a class diagram encoded in XMI format. An interesting aspect of Cranefield's approach is that the UML is used directly and not as a mere visual syntax for another knowledge representation language [10]. He sees the UML/OCL combination as a heavyweight knowledge representation language that is as expressive as or even more expressive than description logic based knowledge representation formalisms.

Another related research initiative examined UML reasoning possibilities by providing a mapping between UML and ontology representation languages. Although this implies that the UML still needs a formal DL-based language foundation to support inference, it also holds that the UML has a mapping to these languages and can therefore be formally defined as equivalent to them. Cali et al. [11] investigated the expressiveness of the class diagram subset of UML in comparison with a description logic. They show that it is possible to apply several description logic inference techniques on standard UML class diagrams. This was achieved by presenting a new formalization of UML class diagrams in terms of the DLR description logic, thereby establishing that UML class diagram semantics can be formally specified. This was

done by mapping the UML class diagram constructs onto the DLR constructs. The DLR is a very expressive Description Logic that is capable of handling n-ary relations (roles), relation intersection and negation (disjointness). The authors focused on the formalization of the class diagram concepts and did not consider OCL constructs. They demonstrated that class diagrams can be given a formal semantics based on description logics. It can therefore be argued that even class diagrams alone are a (at least lightweight) knowledge representation technique which shares many common constructs with description logics.

Other work has focused on a transformation of UML class diagrams into existing semantic web languages like OIL or DAML+OIL [9] to use their inference capabilities and tools for reasoning on UML modeled ontologies. The problem with this approach is that these existing languages are not absolutely equivalent to the UML and the mapping therefore needs to compensate for the differences. Falkovych for example defines a foundational ontology to represent the different mereological types of UML association relationships so that they can be supported in DAML+OIL [9]. A similar approach was taken by Feldering et al. who captured UML aggregation semantics in a general ontology for OIL [9]. On the other hand UML class diagrams do not feature a lot of special OWL axioms (like logical association characteristics, sufficient conditions, etc.) that these approaches cannot represent. In this work the Object Constraint Language was not considered for knowledge representation.

3. UML as a Modeling Syntax for Ontology Representation Languages

The second school of thought focuses on trying to overcome the so called "modeling bottleneck" [9] caused by the lack of professional ontology builders and sophisticated modeling tools for semantic web languages. These approaches see the UML as a modeling syntax for an ontology representation language and do not regard it as possessing any higher knowledge representation and reasoning capabilities. They therefore argue that it needs to be adapted or at least extended to be used for ontology engineering.

The basic idea is to map ontologies represented in this adapted UML into a "real" ontology representation language with a sound description logic foundation. In contrast to the transformation-based approaches in the last subsection which view the UML as a knowledge representation language that only lacks reasoning support and therefore try to map the UML to a formal ontology representation language, this school of thought does not regard the UML as having useful knowledge representation capabilities and see it as a mere concrete syntax for the real knowledge representation and reasoning languages. Kenneth Baclawski et al. [4] support this approach by defining a UML profile to visualize DAML. In this approach UML stereotypes are mapped to DAML elements.

Baclawski et al. propose an extension to the UML metamodel to handle the identified incompatibilities while remaining backward-compatible with existing UML models. The concept of class is perceived as equal in both domains. The foundational concepts in the semantic web languages (Literal, Resource and Thing) are assumed as not existing in the UML and therefore incompatible. This view is untenable, however, since the UML [3] explicitly defines a set of primitive datatypes (String, Integer, Boolean, UnlimitedNatural) and explicitly defines the universal class Object in the MOF [23] and implicitly in the UML specification [3]. The class Object is also well established in implementation models as it is supported by several object-oriented programming languages (e.g. Java, Eiffel). It is therefore a generalization of “everything”. In contrast to “Thing” there is an explicit equivalent to “Nothing” in the OCL namely OclVoid. Baclawski et al. [4] further state that the UML supports non-binary relations that must be reified to be adaptable in DAML (or OWL). If interpreted as the principle that an n-ary relation is reified into a class of objects representing the relation, this argument is correct but it is a general problem of DAML (resp. OWL) and not of the UML. Every time one wants to define an n-ary relation in OWL it has to be "reified" in this manner by creating individuals that represent actual relations and a class that represents the relation on the type level with n binary properties. For different patterns based on this approach consider [24].

The authors state that multiplicity constraints on UML associations affect class memberships [4]. This is in general true as the UML is usually (but not necessarily) interpreted on the basis of the closed-world assumption (CWA). On the other hand it is not clear why open-world assertional knowledge should not be (indirectly) affected by cardinality constraints. They further claim that UML associations have no first class status since their definitions depend on association ends. This assertion is wrong as UML associations are top-level modeling elements that are in the same namespace as classes (i.e., the model or package namespace) and have no obligatory association end type definitions [3]. Baclawski et al. unfortunately confuse first-class status with global scope that is implied by the open-world assumption in OWL. The authors propose to add a new type of model element to the UML metamodel for representing (first-class) properties. These properties are interpreted as the aggregation of association ends from different (and semantically equivalent) associations. The authors unfortunately overlook the UML association redefinition and specialization capabilities that are used to define semantically equal (aggregations of) associations.

Another added metamodel element is Restriction. This is a classifier whose instances are the objects that satisfy a condition on a property associated with the restriction. The Restriction construct is equivalent to usual UML association (end) definitions and specializations and additional OCL statements. A new special construct that is incompatible with the standard UML is therefore unnecessary. The other mentioned incompatibilities regarding cardinality constraints, association taxonomies and namespaces are dubious since they are already challenged and refuted by the authors themselves (see [4]). Baclawski et al. fail to explain how synonyms, sole existential quantifications, the distinction between necessary and sufficient conditions as well as primitive and defined concept descriptions, and the closed-world assumption are handled by their approach. The proposal introduces a set of stereotypes for associations (e.g. TransitiveProperty) and dependencies (e.g. inverseOf, equivalentTo, sameClassAs, samePropertyAs, subPropertyOf) that except for object equivalence can all be represented either directly by a UML construct (e.g. association specialization, bidirectional association) or by an OCL constraint. The approach by Baclawski et al. does not use all UML diagram features and it entirely neglects UML's Object Constraint Language.

The introduced metamodel classes and stereotypes are directly related to DAML constructs, which are largely unknown outside the Semantic Web and DAML communities. The approach can therefore be interpreted as a UML-based graphical syntax for DAML. The authors have developed a UML profile on the basis of the extended metamodel that has already been used in the UML Based Ontology Tool-set (UBOT) project, which is a set of ontology engineering and natural language processing-based text annotation tools [10], and the DAML-UML Enhanced Tool (DUET), which is developed by the Components for Ontology Driven Information Push (CODIP) project and based on Rational Rose add-ins [10]. The benefit (for the DAML community) of this approach is that it enables the use of UML tools like Rational Rose for DAML ontology definition.

3.1 The Ontology Definition Metamodel Approach

The third school of thought holds that the goals of UML and OWL are fundamentally incompatible from one another and can not as a matter of principle be unified. This is depicted in Figure 1 which shows that there is only a relatively small overlap between the feature sets of UML and OWL [12]. This school first argued for a new modeling language to be added to the OMG's metamodel suite which compensates for the handicaps of the UML and provides users with OWL-like ontology representation features within the context of the MOF/MDA infrastructure. The term Ontology Definition Metamodel (ODM) has been coined as a name for the abstract syntax of this language. The focus of the ODM Request for Proposals (RFP) was for a language that allows platform-independent modeling of ontologies. In most of submissions, the proposed ODM is based on a direct mapping of the OWL abstract syntax which makes it more of a platform model. However, as explained in [15], [16], [17], [19] it is the most comprehensive ORL of the Semantic Web .

In its most recent documentation the ODM task force has changed direction however [25]. The group still does not recognize the UML as an ontology definition language but now argues for a family of independent metamodels that represent specific knowledge representation languages in contrast to a single platform-independent metamodel.

Below we introduce two alternative ways of fulfilling the original requirements of the Ontology Definition Metamodel RFP [12] and evaluate their strengths and weaknesses. Original submissions such as IBM's ODM proposal [13] or Gentleware's proposal [14] are either similar to the first proposal (in Section 3.2) or discussed extensively by the University of Karlsruhe's proposal and described along with that proposal in Section 3.3. We are not interested in repeating already available analysis on the first ODM submissions that has been done in the ODM workgroup and in commentaries (e.g. [17]). Therefore we will focus on responses to these submissions and the latest submission of the workgroup. This current ODM proposal is presented in Section 3.4.

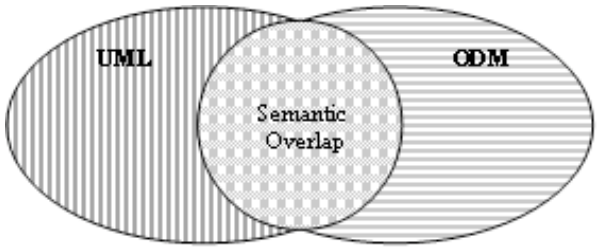


Figure 1 Semantic Overlap between the Modeling and Ontology Representation Languages

3.2 University of Belgrade's Proposal

A team at the University of Belgrade presented a proposal for the ODM and the Ontology UML Profile (OUP) [15], [16]. This proposal is not an official ODM submission but it exemplifies the direction of the first set of proposals (e.g. [13], [19]). In their proposal, the description logic variant of OWL (OWL-DL) is used as the basis for the ODM. The authors highlight the benefits of a MOF-based version of OWL's abstract syntax for the semantic web community [16], and the role of the OUP as a means to utilize the visual modeling capabilities of the UML [15]. The ODM is placed in the second layer of the OMG's four-layer metamodeling hierarchy [16]. Instances of this metamodel

are models in the M1 layer. These ODM models are OWL ontologies which is somewhat contradictory to the ODM's platform-independent role in the ODM RFP [12]. The problem with this approach is that the ODM has no concrete syntax other than the XMI representation of its models. Therefore, the approach makes use of another MOF-based modeling language namely the UML.

To use the UML concrete syntax in an adapted form for the ODM, the ODM RFP suggests the workaround of a UML profile. This profile uses the stereotype mechanism to introduce ODM constructs into the UML metamodel so that M1 models of this profile can instantiate these constructs. The OUP models can then be transformed via XSLT into ODM XMI or even OWL XML representations. By using the profile mechanism, the authors avoid the consequences of a real extension of the UML metamodel which would require the new constructs to be direct subclasses of the UML metaclasses [15]. The stereotype annotations to the model can be applied in a much more arbitrary and ad hoc style. The models created with the profile do not need to be consistent UML models. In fact the OUP models are effectively OWL ontologies in a UML like syntax, which is exactly what the authors intended.

The problem with this approach is that it seems to be only loosely based on the UML. The only resemblance it bears to the UML is that OUP model elements that are interpreted as sets of individuals or of tuples of individuals are depicted as classes. However, while associations and attributes in a UML model are interpreted as associations in the semantic domain, OUP associations and attributes are interpreted as meta-attributes. The single OUP constructs have no interpretation in the semantic domain; they are only dummy metamodel elements that allow certain OWL expressions to be expressed in a UML diagram. This is the reason why the authors allude to the OUP as not being a stand-alone ontology language [15]. Otherwise, one would wonder why two metamodels are needed for the same purpose. However, since OUP elements such as restrictions, allDifferent and the anonymous class descriptions of the ODM have no UML compatible class interpretation in the semantic domain, the OUP cannot be understood as a language but only as a vocabulary of annotations of rectangles and lines using the UML profile mechanism. Nevertheless, basing a new language's concrete syntax on another language's syntax by adapting this language's abstract syntax is awkward. The plan is to provide an interim solution until a separate syntax for the ODM is defined so that only transformations from the ODM to the OWL are needed. To what extent the new syntax differs from the UML syntax is an open issue.

3.3 University of Karlsruhe's Proposal

Like the proposal described in the last subsection this proposal also proposes an ODM for OWL DL [17]. The difference between this and the other proposals is that the authors explicitly try to achieve an intuitive notation for both communities [17]. This means that the result is not intended to just be a new modeling syntax for OWL but also a chance for UML users to easily model OWL ontologies. The proposal therefore stands in contrast to most of the other ODM proposals. The authors view the IBM proposal [13] as being extremely counterintuitive as it models OWL properties as UML classes and OWL class constructors as UML associations [17]. Gentleware's proposal [14] is also criticized as being a cumbersome RDF serialization of the OWL. This proposal also uses UML classes to depict OWL properties and comments to specify specific class constructors. Sandpiper's [20] and the DSTC's [18] approaches are criticized for providing no visual syntax or metamodel. The merged proposal [19] is criticized for having too wide a scope which leads to complex mappings and bad readability and usability [17]. The authors therefore argue that a metamodel for every KR paradigm and language should be developed. This is in fact the role of platform specific metamodels. However, this is not the role of the ODM, which is explicitly designated to be platform independent. Their proposal is not really an ODM proposal therefore but a proposal for a platform specific model of OWL DL.

Like every other proposal in the context of the ODM RFP this proposal ignores the OCL and only tries to map OWL DL constructs to existing or customized UML diagrammatic constructs. They therefore explicitly introduce equivalentProperty and equivalentClass associations into their ODM metamodel [17]. The omission of OCL constraints leads to the requirement for graphical representations for all metamodel constructs. These are defined in the author's UML-profile for ontologies. New constructs are needed for anonymous class descriptions in the context of logical and property restriction class constructors. Class equivalence is depicted by a bidirectional generalization arrow [17]. OWL object properties are represented as UML n-ary associations (i.e., the diamond notation); while datatype properties are depicted as UML attributes [17]. This ignores UML's ability to redefine associations as explained in [3]. It is also problematic because property restrictions are erroneously depicted as domain and range values for the property, which is not intended by OWL. UML attributes cannot be specialized and have no classifier interpretation in the UML. It is therefore difficult to model datatype properties only in this way. Their enumerated datatype construct [17] is completely redundant as the UML features exactly this concept as well (cf. [3]). This proposal uses the UML instance specification construct to depict individuals. In contrast to IBM's proposal, which suggests using a class Thing for individuals, they argue that this would blur the difference between object and model level in the UML. The actual problem is not a universal concept Thing, but the distinction between Thing being the logical classifier of all

individuals and Object or InstanceSpecification being their linguistic classifier. A two dimensional infrastructure [5] explicitly requires a universal concept Thing as is further explained in [21].

3.4 The ODM Task Force's Family of Metamodels Proposal

The ODM workgroup recently changed their initial approach of a single language- and formalism-independent ontology metamodel with an associated hub-and-spoke architecture, where the ODM would be the central metamodel into which every ontology in any language would be translated (and vice versa). The new ODM is a collection of language-specific metamodels for several existing knowledge representation languages. Currently supported are the Simple Common Logic [27], Topic Maps [26], RDFS, OWL Full and Entity Relationship modeling. An ontology in one of these languages can be transformed into an ODM model and thus available within the MDA infrastructure without loss of information. This was not possible with the original approach of a single general ODM metamodel. A set of mappings between the metamodels is defined to support interoperability. To reduce the number of transformation mappings OWL Full has been selected as a central hub metamodel. The mappings should support the use of legacy models as a starting point for ontology development and they should enable users to choose from the range of different languages based on the required degree of expressivity and change as needed [25]. A third major component of the new ODM proposal is the collection of UML profiles for the different knowledge representation metamodels (currently RDFS, OWL and Topic Maps). These profiles should provide a bridge between the UML and knowledge representation communities [25], or more precisely between UML tools and the KR communities. Models and ontologies are bridged by the appropriate bidirectional mapping between OWL Full and UML. The profiles therefore serve as a means to utilize UML modeling tools for visual ontology development. The UML profile for OWL suffers from the same problems as identified in [17] as it is based on the IBM proposal [13]. Finally, in order to respect the OMG four-layer metamodel hierarchy, a foundational ontology has been added for RDFS and OWL that encompasses the OWL/RDFS elements that belong to the model level (i.e. owl:Thing, owl:Nothing, rdf:nil, and the XML Schema datatypes) as an M1 model library. The new ODM approach is a mixture of the old hub-and-spoke architecture, which is now based on OWL Full (as depicted in Figure 2), and the second school of thought (Section 0) that used the UML as a mere modeling syntax.

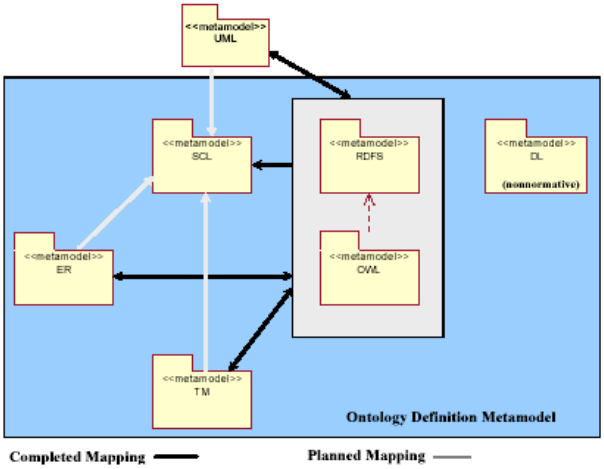


Figure 2 Structure of the ODM from [25], Figure 1, Section 9, page 74

The new ODM proposal is based on the premise that the UML is not a practical language for representing ontologies. This premise is justified by a comparison between the UML and the OWL Full languages [25], [2] which is claimed to identify the incompatibilities between the two languages. However, it rather reveals a tremendous overlap between the two languages. Only a few concepts needed for ontology representation are not available in the UML class diagram metamodel. Moreover, the authors unfortunately omit the OCL from the comparison and thus do not consider the possibility that the OCL might feature the missing OWL constructs. In fact, the authors state that the OCL is a predicate definition language that is more expressive than OWL Full [25]. The analysis in [25] identified the following potential shortcomings of the UML: support for synonyms, extension equivalence specification, sufficient conditions, complex class constructors, the logical characteristics of associations, existential quantification and value restriction, global properties, autonomous individuals, class-specific cardinality constraints, the universal concept Thing and the OWL Full feature of classes as instances. However, most of these potential shortcomings have been disproved in [21], which we will briefly summarize here. The universal concept is part of the OCL and implicitly also of the UML. It is no problem to assume its existence or make it explicitly available. Value restriction is implicitly applied to every association as the UML follows the CWA. UML associations are as global as global properties in the OWL. An absolute global (or universal) scope, however, has to be explicitly specified. Local restrictions are made by association redefinitions or specializations which includes class-specific cardinality constraints. The intended meaning of "global" in the comparison [25], however, is that properties in OWL can apply to every class and have universal scope. This is related to the open-world

assumption and can be resembled by defining every association in the context of the universal class. In the UML, under certain assumptions all individuals (objects) are autonomous, otherwise multiple and dynamic classification would not be feasible. It is even possible for an object to be only an instance of the universal concept. All subclasses are by default defined to form a non-covering partitioning of the universal class. Sufficient conditions, complex class constructors, extension equivalence, logical characteristics of associations and existential quantification restrictions are all realizable with the OCL. Some of them are even realizable with UML diagram elements using anonymous classes and generalization-set constraints. The definition of metaclasses as in OWL Full is done using the UML stereotype mechanism. Nevertheless, a more appropriate solution for both languages would be a two dimensional modeling infrastructure as explained in [5], [21]. The only real missing feature identified by the analysis are synonyms that relate to the unique name assumption in the UML which is omitted in the OWL. However, the authors rather state that the UML places no constraints on names at the M0 level [25]. This is correct but it has no influence on the fact that the representation of M0 level elements in UML object models (M1) underlies the unique name assumption. On the other hand, UML datatype enumerations are spuriously equated with OWL object enumerations [25] and the consequences of closed-world semantics in the extensional knowledge are disregarded.

4 Summary of the Approaches

In addition to the three widely recognized approaches described above, three further options can be selected. One is to use an existing IRL. Another is to develop a new IRL that provides the required characteristics. And a third way is to adapt an existing IRL to the requirements. The first option can be discarded as no IRL fulfills the needs. The second option has been effectively disapproved by the ODM task force. The OMG ODM RFP and the first submissions believed in the creation of a new metamodel for ontology definition. That metamodel should encompass all relevant IRLs. However, the recent proposals show that a new metamodel was neither feasible due to insufficient commonalities nor wanted as existing IRL could also perform the job. The presented approaches differ in the selected IRL and the taken action to meet the needed requirements. The major candidates for an IRL are the UML and OWL. The UML posses the needed user-friendly interface, a wide tool support and a large educated work force. However, it lacks a formal semantics. OWL on the other hand possesses a model theoretic semantics and with the DL sublanguage a deducible set of constructs. On the other hand lacks OWL a user-friendly interface, broad tool support and experienced users.

The first approach points out that the UML can be mapped to a formal ORL, which implies that a formal semantics can be applied to the UML. However, the approach to date lacks a complete definition of the UML/OCL semantics and is unclear on how the UML could be OWL compatible, i.e. define OWL compliant ontologies. The second approach suffers from being somewhat isolated because of a single ORL approach without a metamodel and MDA-based transformation support, and inadequate usability due to a direct representation of the ORL's abstract syntax in a UML profile.

In the approaches that base the ODM completely on OWL, the ODM is a platform-specific metamodel for OWL ontology definition. The advantage of this approach is that MDA tools can be used for model management and transformation of an ODM model into a model of another MOF-based language, according to a transformation definition. Moreover, like all MOF-based languages the ODM supports an XMI serialization that allows tool interoperability. The UML profiles enable the use of existing UML tools, which right away become ontology modeling tools. These benefits are balanced by the problems that the direct mapping of OWL constructs into a modeling language brings. Several OWL constructs like restrictions cannot be well represented visually. Their graphical presentation is very awkward and results in complex and clumsy models. This results in reduced usefulness and productivity. The use of OWL terminology leads to a lot of synonyms and homonyms for UML constructs that cause confusion. A business engineer who is not familiar with the Web Ontology Language is not likely to be familiar with the UML profile for OWL. An experienced ontology engineer on the other hand would have to handle the specialties and pitfalls of the UML. Although we identified several issues where the University of Karlsruhe's proposal (Section 3.3) does not use original UML constructs (especially the complete OCL), we believe this is the best attempt so far (of the ODM proposals) to match UML constructs with OWL constructs and thus to increase the usefulness of such models. A critical issue is the difference between the UML closed-world assumption and OWL's open-world assumption. An ODM-based approach that only uses the UML concrete syntax only has to frame rules for its correct open-world semantics usage, although this might be a problem for experienced UML users. However, a profile based approach is technically bound to the UML semantics. Only the current ODM submission partly addresses this issue by adding a foundational ontology.

The current ODM approach facilitates interoperability among IRLs. It allows UML models to be translated into an OWL Full model and afterwards into any other ORL ontology. However, the UML to OWL mapping guaranties only lightweight ontology modeling support for the UML, otherwise the UML would have been used as the ODM in

Table 1 Comparison of the presented approaches with their characteristics

The approaches and their characteristics and feature fulfillment		Position of approach in paper (section reference)	Usability	Supported IRLs and interoperability	Use of UML-tool and language infrastructure	Utilization of MOF and MDA infrastructure	Easily translatable into (Semantic Web) ORLs	Instant access for business and application developers	Instant access for ontology engineers	Large experienced workforce	Access to OE-technology for modeling community	Access to MDA-technology for ontology engineering community
UML+OCL as ORL		2	+	-	+	+	o	+	-	+	+	o
UML-profile as modeling syntax		3	-	-	+	-	+	-	o	-	-	-
MOF-based abstract syntax and UML-based	Old ODM	4.1	-	o	+	+	+	-	+	-	-	o
	UML + OWL-Mix	4.2	o	o	+	+	+	o	o	o	o	+
	New ODM	4.3	o	+	+	+	+	-	o	o	+	+

Legend:
+ fully achieved, - not achieved, o partly achieved

the first place. The special role of the OWL metamodel in the ODM reinforces the gap between the two technology spaces and compounds the confusion within the MDA framework as it tries to establish another general purpose modeling language in the MDA architecture. This reinforces the message that there is a fundamental difference between system, knowledge and data representation.

Neglecting the OCL forces the representation of all OWL DL constructs as graphical constructs. The problem of this approach lies in the characteristics of graphical representation languages. As was already realized in the OO-modeling community it is not possible or favorable to represent every logical construct in a graphical modeling language. The important disadvantage of a visual language is that some logical statements, which could be easily and concisely defined in textual languages, get very complex and cumbersome in a visual language. The usability/complexity trade-off which makes visual languages easy to comprehend in general but clumsy for representing complex expressions led the OO-modeling community to develop the OCL as an equal companion of the UML. Unfortunately, only one approach to ontology representation using the UML has so far taken the OCL into consideration.

A few of the different characteristics of the presented approaches are depicted in Table 1. The initial requirement was a user-friendly language that can be easily translated into ORLs. Both characteristics can be read off the table. The second is clear for a language that is already an ORL. How the UML can be translated into OWL and other ORLs is not completely explained. Usability, on the other hand, is composed of several sub-properties (homogeneity of language environment, instant access, deflection from familiar standards, efficiency of modeling, etc.).

5 Conclusion

In this paper we have characterized the space of possible strategies for integrating ontology representation and modeling technologies. We have identified the three major schools of thought on how this should be achieved, and have analyzed all the existing proposals to a reasonable level of detail.

This investigation has revealed that contrary to popular belief UML/OCL is a potential candidate for use as a higher-level information representation language for enterprise information. In fact UML/OCL is as good as an ORL for supporting the semantically precise representation of information provided that the appropriate frame

conditions are defined using OCL constraints. Therefore, given the very high number of educated UML designers who already use it for requirements engineering and business process design, as well as the ease of use and broad tool support for the language it makes sense to use the UML as the basis for the standard information representation language.

While the ODM claims that the UML is only needed for the support of legacy information [25], we believe that the UML as a platform-independent metamodel should have the primary role in ontology modeling. The different ODM metamodels serve as platform-specific metamodels, and mappings between the UML and these metamodels act as MDA-compliant transformation rules. The UML profiles will not be the primary ontology development syntax for the end user but an intermediate format for specialized MDA engineers. The user can still choose between different expressivity and computability trade-offs, but will only need to model once in the UML.

A fair compromise would be the parallel existence of both approaches. This would afford the same bidirectional mappings to and from the UML as they are currently available for OWL Full. The user could then choose which language – the UML or the UML profile for OWL – that he or she prefers for platform-independent ontology modeling.

The problem with the UML/OCL framework at the moment is that it is too powerful (i.e. too expressive) rather than too weak for the purpose of creating semantically precise and computational representations of enterprise information. What is needed, therefore, is a way of tailoring the frame assumptions that bound the semantics of UML models so that, where needed, they can be restricted to the semantics of Description Logics which have desirable computational properties. However, this does not imply the need for disjoint sets of features or complex mappings between visual but semantically vacuous pictures of enterprise information and concrete but user inaccessible representations. On the contrary, it implies the need for a single unified core set of modeling features supported by tailorable frame assumptions [22].

Thus, to conclude, we firmly believe that the approach advocated by the first school of thought is the correct one in the long run, that the current ODM proposal is an important part of the final solution, and that the other approaches, while representing helpful bridging technologies, should not form the basis for a future unified enterprise information representation language.

References

[1] Michael K. Smith, Chris Welty, and Deborah L. McGuinness, Editors, "OWL Web Ontology Language Guide", *W3C Recommendation*, 10 February 2004, http://www.w3.org/TR/2004/REC-owl-guide-20040210/, [30 Mai 2005]. Latest version available at http://www.w3.org/TR/owl-guide/.

[2] Lewis Hart, Patrick Emery, Bob Comb, Kerry Raymond, Sarah Taraporewalla, Dan Chang, Yiming Ye, Elisa Kendall, Mark Dutra, "OWL Full and UML 2.0 Compared", *OMG TFC Report*, 2004.

[3] Object Management Group, "UML 2.0 Superstructure Revised Final Adopted Specification", *OMG Specification*, October 2004.

[4] Kenneth Baclawski, Mieczyslaw M. Kokar, Paul A. Kogut, Lewis Hart, Jeffrey E. Smith, Jerzy Letkowski, and Pat Emery, "Extending the Unified Modeling Language for ontology development", *Software and System Modeling*, vol. 1 , pages 142-156, 2002.

[5] Colin Atkinson and Thomas Kühne, "Model-Driven Development: A Metamodeling Foundation", *IEEE Software*, vol. 20, pages 36-41, 2003.

[6] Warmer, J., Kleppe, A., "The Object Constraint Language, Second Edition. Getting your Model ready for MDA", Addison Wesley, 2003.

[7] Stephen Cranefield and Martin Purvis, "UML as an Ontology Modelling Language", in Proceedings of the *IJCAI-99 Workshop on Intelligent Information Integration*, held on July 31, 1999 in conjunction with the Sixteenth International Joint Conference on Artificial Intelligence City Conference Center, Stockholm, Sweden, 1999.

[8] Stephen Cranefield, "UML and the Semantic Web", in Proceedings of *SWWS'01, The first Semantic Web Working Symposium*, Stanford University, California, USA, pages 113-130, 2001.

[9] Kateryna Falkovych, Marta Sabou, and Heiner Stuckenschmidt, "UML for the Semantic Web: Transformation-Based Approaches", in Omelayenko, B. and Klein, M. eds. *Knowledge Transformation for the Semantic Web, Frontiers in Artificial Intelligence and Applications*, vol. 95, IOS Press, pages 92-106, 2003.

[10] Paul Kogut, Stephen Cranefield, Lewis Hart, Mark Dutra, Kenneth Baclawski and Jerry Smith, "UML for Ontology Development". *Knowledge Engineering Journal*, vol. 17, issue 1, pages 61-64, March 2002.

[11] Andrea Cali, Diego Calvanese, Giuseppe De Giacomo and Maurizio Lenzerini, "A Formal Framework for Reasoning on UML Class Diagrams", *Lecture Notes in Computer Science*, vol. 2366, pages 503-512, 2002.

[12] Object Management Group, "Ontology Definition Metamodel Request for Proposal", *OMG RFP*, 2003.

[13] IBM, "Ontology Definition Metamodel (ODM) Proposal". August, 2003.

[14] Gentleware, "Ontology Definition Meta-Model". August, 2003.

[15] Dragan Djuric, "MDA-based Ontology Infrastructure". Comput. Sci. Inf. Syst., vol. 1, 2004

[16] Dragan Djuric, Dragan Gasevic, and Vladan Devedzic, "Ontology Modeling and MDA." Journal of Object Technology, vol. 4, no. 1, pages 109-128, 2005.

[17] Brockmans, S., Volz, R., Eberhart, A., and Löffler, P. (2004), "Visual Modeling of OWL DL Ontologies Using UML", in *International Semantic Web Conference*, pages 198-213, 2004.

[18] DSTC, "Ontology Definition MetaModel Initial Submission". August 2003.

[19] Lewis Hart, Patrick Emery, Bob Comb, Kerry Raymond, Dan Chang, Yiming Ye, Elisa Kendall, Mark Dutra, "Usage Scenarios and Goals For Ontology Definition Metamodel", URL: http://www.omg.org/docs/ontology/04-01-01.pdf. January 2004.

[20] Sandpiper Software Inc. and Stanford University Knowledge Systems Laboratory, "UML for Knowledge Representation. A Layered, Component-Based Approach to Ontology Development", URL: http://www.omg.org/docs/ad/03-08-06.pdf. March 2003.

[21] Kilian Kiko, "Towards a Unified Knowledge Representation Framework," *Master Thesis*, 2005.

[22] Colin Atkinson, "Unifying MDA and Knowledge Representation Technologies", *The Model-Driven Semantic Web Workshop (MDSW 2004)*, September, Monterey CA 2004.

[23] Object Management Group, "Meta Object Facility 2.0 Core Specification", *OMG Specification*, October 2003.

[24] Natasha Noy, Alan Rector, "Defining N-ary Relations on the Semantic Web: Use With Individuals", *W3C Working Draft,* 21 July 2004, http://www.w3.org/TR/swbp-n-aryRelations/ [20 August 2005].

[25] Object Management Group, DSTC, IBM, Sandpiper Software Inc., "Ontology Definition Metamodel Second Revised Submission", May 2005.

[26] ISO/IEC 13250-2, "Topic Maps – Data Model". Latest version is available at http://www.isotopicmaps.org/sam/sam-model/.

[27] ISO/IEC WD 24707 Information technology, "Common Logic (Common Logic) – A Framework for a Family of Logic-Based Languages", 2005-03-11. Latest version is available at http://cl.tamu.edu/

Enterprise Architecture Framework based on MDA to Support Virtual Enterprise Modeling

[1]Tae-Young Kim, [2]Cheol-Han Kim, [3]Jeong-Soo Lee, and [4]Kwangsoo Kim
[1,3,4]*Dept. of Industrial and Management Engineering,*
Pohang University of Science and Technology, Republic of Korea
[2]*Division of Information Communications and Internet Engineering,*
Daejeon University, Republic of Korea
{[1]west, [3]jsrhyme, [4]kskim}@postech.ac.kr, [2]chkim@dju.ac.kr

Abstract

Unfortunately, there is no previous approach to fully support the designing and the managing of Virtual Enterprise (VE) in an elegant manner, because of the diversity and turbulency of the business environments. This observation has motivated this research to develop an integrated systematic framework by harmonizing several approaches such as Enterprise Architecture (EA), framework-based development, Model Driven Architecture (MDA), and meta-modeling approach, etc.

Accordingly, the issue of this research is to suggest an enterprise architecture framework based on MDA to contribute to the configuration of the VE. As the MDA approach is similar to the business formation of the VE: business scenario design, business process design for business logic, and functional design for execution, this framework can be used for business managers or business domain experts to build the collaborative VE models quickly and effectively with insights.

1. Introduction

Today, enterprises are facing a rapidly changing business environment and can no longer make predictable long term provisions, because of the turbulent market conditions, regulations of the working conditions, fast technological mutation, and so on. And the business competition is no longer enterprise to enterprise, but value chain to value chain such as design chain, supply chain, and customer chain. This requires not only intra-integration of an enterprise but also inter-integration among business partners to make seamless business processes. This also makes enterprises focus only their core capabilities on their value chain, while they collaborate with other enterprises that have other complementary capabilities. As each enterprise operates as a node in the network that is composed of suppliers, customers, engineers, and other specialized service providers, the process-centric loosely-coupled integration focusing on the optimization of the value chain is emerging as a result [1].

Therefore, new enterprise models which support process-centric loosely-coupled integration are needed. These models can be derived from the latest approaches such as Enterprise Architecture (EA), framework-based development, Model Driven Architecture (MDA), and meta-modeling approaches. Using these approaches, the value chain can be combined dynamically and optionally through “plug & play” way on the business environment. Moreover, if the buzz information technology, namely Service Oriented Architecture (SOA), is grafted together, the complex and dynamic business process of the Virtual Enterprise (VE) can be considered as a set of service components in order to support the collaborative business processes.

There have been, however, some critical problems in the realization of the VE so far. Firstly, the real business processes of the VE are very context-dependent and complex. So, it is not suitable to apply the traditional business process methodology. Secondly, the VE has multiple stakeholders who are interested in different aspects of the enterprise models. But there is no comprehensive modeling approach to support diverse modeling component required by the stakeholders. Thirdly, the business processes of VE are

very distributed and heterogeneous across the value chain. However, there is a lack of standard definitions and effective mechanisms which guarantee the interoperability of the enterprise models.

Unfortunately, so far there is no approach that fully solves these important problems in the realization of the VE. This research suggests a new systematic approach harmonizing above mentioned approaches. It is a coherent enterprise modeling framework that underpins the representation of enterprise models from different viewpoints, at different levels of granularity, generality and abstraction. This framework provides insights, enables communication among stakeholders and guides complicated change processes. This framework is expected to significantly contribute to the configuration of the VE.

The rest of this paper is organized as follows: Section 2 describes previous works related to the VE configuration framework. Section 3 explains the suggested VE configuration framework in detail. Finally, section 4 provides conclusions and future works.

2. Related Works

Although there are a lot of interests in enterprise engineering for designing the enterprise models, there is no well-established common methodology to completely support the agility and the interoperability of the VE models. This research considers that the following relevant approaches can play important roles in developing a systematic framework for designing the agile and interoperable enterprise models.

2.1. Business Process-centric Architecture

According to Smith and Fingar, the business process management will be the heart of the future business systems and will support the dynamic integration and collaboration of all participants in the value chain [2]. In the context of this research, the VE is more focusing on what can be done to achieve the common goals. And the VE should be more loosely-coupled the value chains based on the collaborative business processes between business partners. The collaborative business process is defined as a set of linked activities that are distributed at business partners of the VE [3]. These business processes should be managed and controlled autonomously in a distributed environment, because each business partners can be in different places and different time zones. Accordingly, the foundation of the VE is the process-centric enterprise integration approach that is supported by the distributed business process management.

2.2. Enterprise Architecture (EA)

The term “Enterprise Architecture” refers to a comprehensive description of all of the key elements and their relationships that make up an organization [4]. The EA identifies the essential processes performed by the VE, shows how the VE performs these processes, and also includes methodologies for the configuration of the VE [5]. These features enable business managers to understand how their enterprise models are doing and to make decisions about changes that lead to appropriate modification in response to business environments.

The earliest systematic framework that we know as the enterprise architecture framework is Zachman framework [6]. The key idea of Zachman framework is that an overall architecture is made up of a number of other architectural components that are focusing on different, specific areas of concern [7]. Several U.S. federal departments have developed their own EA based on Zachman framework: Federal Enterprise Architecture Framework (FEAF), Department of Defense Enterprise Architecture Framework (DoDAF) and so on [8][9]. The most extensive efforts up-to-date in the development of reference architecture for a single enterprise have been undertaken by Generalised Enterprise Reference Architecture and Methodology (GERAM) [10]. GERAM includes the harmonization with software engineering, system engineering, developments of frameworks, and the researches on PERA, CIMOSA, GRAI-GIM, etc. which are reference architectures to organize all enterprise integration knowledge and serve as a guide in enterprise integration programs.

2.3. Framework-based Development

The framework-based development is usually said to be 2nd generation business process methodology [11]. Important efforts are dedicated to exploiting best practices and design patterns of the business processes, the business components, and the architectural frameworks for the reusable sets of coherent design and implementation.

Supply Chain Council (SCC) established Supply Chain Operations Reference Model (SCOR) for the supply chain management domain [12]. In the domain of tele-communication, Next Generation Operations Systems and Software (NGOSS) was proposed by TeleManagement Forum (TMF) [13][14]. And the Instrumentation, Systems, and Automation Society (ISA) has tried to standardize the manufacturing processes in Manufacturing Execution System (MES) domain [15].

As many quality properties such as maintainability, portability, efficiency, reusability, etc., rely on the framework-based development way, this is essential to design the agile and interoperable enterprise models and its loosely-coupled integration. Consequently, the framework-based development allows improving and accelerating the development of the VE models.

2.4. Model Driven Architecture (MDA)

As the technology platforms of the enterprise systems continue changing quickly and the demands of integrating existing heterogeneous legacy systems continue growing increasingly, new modeling paradigm, namely MDA, has created a buzz of interest by promising to increase the productivity, the flexibility, and the portability of the enterprise models.

The MDA, which is an initiative by the OMG, has a strong correlation to the concepts of the SOA in a more abstract level. As the MDA makes a distinction between Platform Independent Model (PIM) and Platform Specific Model (PSM), it provides an open, technology-neutral approach to the challenge of business and technology change [16].

The concepts of the MDA can be used in designing the VE models because the VE formation is established through "business scenario design", "business process design", and "business function design" after finding business opportunity [17]. The MDA can be used by VE brokers who want to design VE models quickly and effectively with insights.

2.5. Meta-modeling Approach

As "meta-model" means the rendering of a language definition, the meta-modeling approach is becoming a standard way of defining and managing the meta-models for representing the enterprise models. Therefore the meta-modeling approach can be used for enabling the designed models and the defined meta-models to have the interoperability with each other.

Recently, it has been demonstrated that the meta-modeling can define concrete syntax and abstract syntax, as well as semantics [18][19][20]. OMG has suggested UML profiles in the form of the extended UML meta-model to make good use on particular domains. And it has been discussed how a UML profile can be defined for a specific domain that requires a specialization of the general UML meta-model in order to enable the UML to more precisely describe the domain [20]. The MDA defines the 4-layer architecture for structuring this meta-modeling.

These up-to-date approaches have their own advantages, but they are dealt in separate domains and developed independently to address their own purposes. There is not a common framework that can integrate these approaches in order to configure the agile and interoperable VE. This observation has motivated this research to develop the enterprise architecture framework based on MDA to support enterprise modeling by integrating together the advantages of all the discussed approaches.

3. Enterprise Architecture Framework based on MDA

In order to configure the VE, the proposed overall framework is illustrated in figure 1. The left-side of figure 1 shows the process of the enterprise configuration. It contains 4 phases focusing on details of the enterprise configuration. The right-side of figure 1 briefly shows the facilities to support each phase.

Each designing system such as *EA Designer*, *Meta-model Designer*, *CIM/PIM Modeler* and *PSM Mapper* is associated with each relevant reference repository, local instance repository, and ontology repository. To increase portability, efficiency, agility and interoperability of the models, each system reuses the best practices stored in each reference repository under the concept of the framework-based development. To support the communication and the comprehension for retrieval and use, each system is also connected with ontology repository.

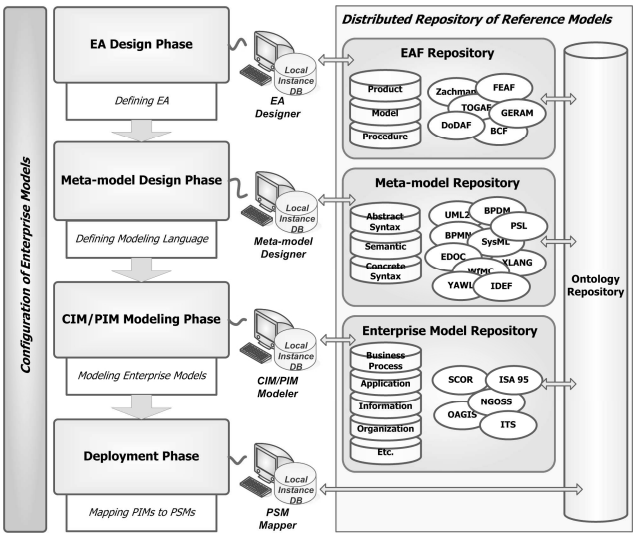


Figure 1. Overall framework

The following sections from 3.1 to 3.4 explain each phase of the framework in detail.

3.1. EA Design Phase

First of all, the business partners of the VE establish a standardized collaborative model through the EA which defines all the elements and the perspectives of the enterprise models and explains how they work together as a whole. It is important that the formal EA specification should ideally capture all the aspects that are unique to the enterprise system, and also help in reasoning various architecture decisions.

The procedure to establish the EA is described as following:

- *Determine the organizational structure of the VE*: The organizational structure of the VE means the business partners which take part into the business processes of the VE.
- *Assign the roles of each business partners*: The roles should be assigned to the selected major business partners
- *Decompose Views and Perspectives*: Because the business components and the business processes, in a broad sense, can be understood from a number of different views at different abstract levels, the process of decomposing and separating concerns of various participants should be performed.
- *Establish the EA*: Through the above process of decomposing concerns of various participants, the EA can be built which supports the different views and the different abstract levels. According to the framework-based development, a new EA can be created rapidly and efficiently with reusing the best practices of the existing EAs.
- *Fulfill the EA*: Once we have established the EA, we begin to collect reference models that fulfill the EA, such as business process models, information models, resource models, etc., from the suitable reference repositories.

The meta-model of the EA in this research is constructed as figure 2. Basically, the EA has several perspective layers and some views. In this research, there are two assumptions: One is that each perspective of the EA refers to the models of the MDA such as Computation Independent Model (CIM), PIM, and PSM [21]. The other is that each view of the EA corresponds to one modeling domain in order to support the different stakeholders such as business process manager, information system manager, organization manager, etc. who have different concerns and methodologies.

A pair of one perspective and one view produces an EA cell. And each EA cell provides a container for the cell content list, the modeling language, and some enterprise models and enterprise reference models.

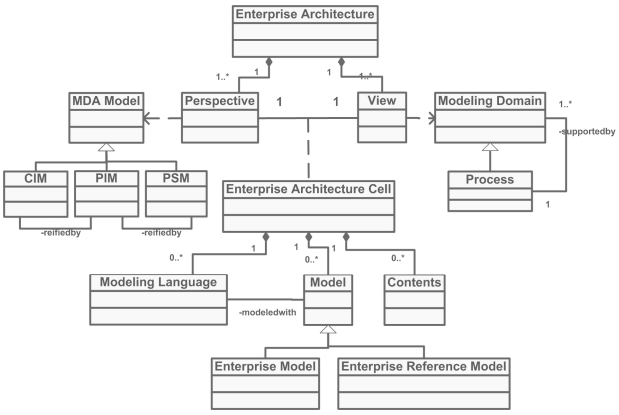


Figure 2. Meta-model of EA

While the business process is not originally the central part of Zachman framework, the design of the EA is based on the business process-centric architecture in this research.

Consequently, we divide the column (views) of the EA into 5 modeling domains in this research.

- *Process Domain*: Focusing on the business processes of the VE
- *Application Domain*: Focusing on the business applications which support the business processes
- *Information Domain*: Focusing on the business information or system information which supports the business processes and applications respectively
- *Organization Domain*: Focusing on the participants who are responsible for the support and execution of the business processes
- *Technology Domain*: Focusing on the technology environment and infrastructures which support the business applications

Meanwhile, the rows of the EA are made of the different perspectives. The perspectives are the contextual layer, the conceptual layer, the logical layer and the physical layer in this research.

- *Contextual Layer*: Defining the goal, purpose and visions of the VE, which are restricting the business boundary. Regarded as CIM in the MDA
- *Conceptual Layer*: Defining the models of value chains of the VE in business terms, which are including its business processes, business partners, etc. Regarded as CIM in the MDA
- *Logical Layer*: Defining the models of the business processes and the business components in more rigorous terms than the conceptual Layer. Regarded as PIM in the MDA

- *Physical Layer*: Defining the deployed models related with specific technology platforms. Regarded as PSM in the MDA

Figure 3 illustrates *EA Designer* for the EA design phase. A designed EA example is displayed in the main screen of *EA Designer* and each EA cell provides room for the contents list, the meta-model, the enterprise models, and the enterprise reference models.

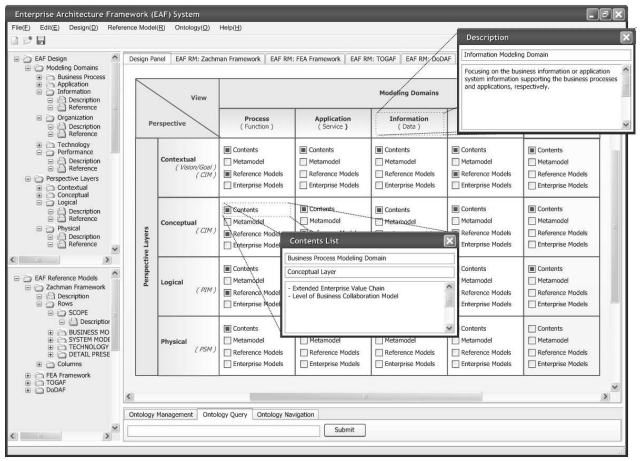


Figure 3. EA Designer

3.2. Meta-model Design Phase

In this phase, modeling languages are developed to describe the enterprise models which are comprised in each cell of the EA.

Of course, some generic purpose modeling languages such as UML or IDEF can be used. However, when we are going to make it easier for a domain expert to solve problems using models, it is very important that the modeling language can clearly represent the problem domain [22]. In order to achieve more effective and correct modeling capability, this research tries to bring in the idea of Domain Specific Methodology (DSM), and also tries to design the specialized modeling languages for each business domain.

Standing on the basis of the meta-modeling approach, the specialized modeling languages can be designed as the meta-models. To support it, this research provides a combined architecture of the MDA's 4-layer meta-modeling architecture and the EA, as shown in figure 4.

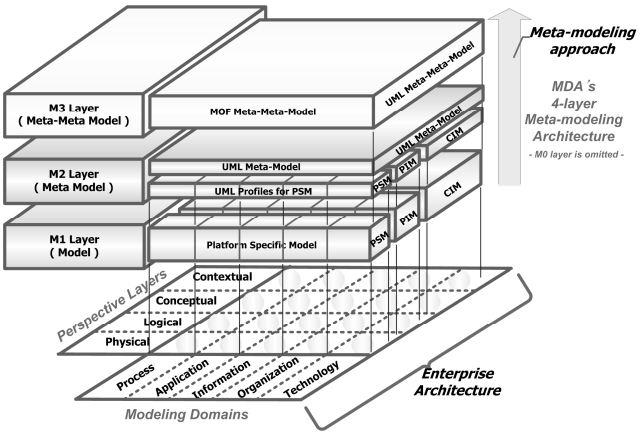


Figure 4. Combined architecture of MDA and EA

As mentioned above, the contextual layer and the conceptual layers of the EA correspond to CIM of the MDA, and the logical and the physical layers correspond to PIM and PSM of the MDA, respectively. On the basis of this, CIM, PIM and PSM at M1(model) layer of the MDA are built in our EA, as illustrated in figure 4. Above M1(model) layer, there is M2(meta-model) layer where the modeling languages, i.e. the meta-models, are defined to describe each model. Above M2(meta-model) layer, there is M3(meta-meta-model) layer which is the top layer of the 4-layer meta-modeling architecture of the MDA.

Because the UML extension mechanism of UML profile appears to be very useful to define a suite of the modeling languages, our meta-models are developed in the form of UML profiles, which is MOF-compliant, at M2(meta-model) layer.

The internal architecture of UML profiles is given by figure 5. A language definition comprises an abstract syntax, semantics, and any number of concrete syntaxes [20]. The abstract syntax is a model of the valid expressions of the language, which is abstracted away from any particular concrete rendition of those expressions. There may be several concrete syntaxes for one abstract syntax. Semantics concerns the definition of what it means.

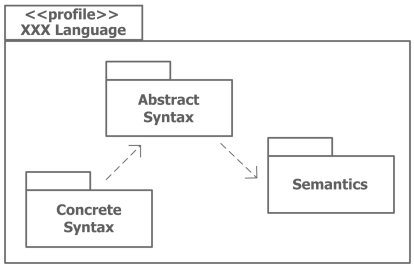


Figure 5. Modeling language

Figure 6 illustrates *Meta-model Designer* for the meta-model design phase and a designed meta-model example.

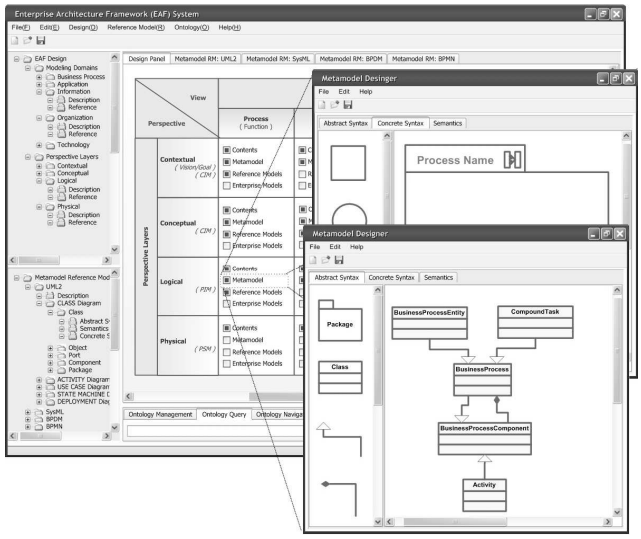


Figure 6. Meta-model Designer

As shown above, the phase of designing the meta-models makes progress on the EA. The designed elements of the meta-model such as abstract syntax, concrete syntax, and semantics are packed in the EA.

3.3. CIM/PIM Modeling Phase

The modeling process for the VE proceeds on each perspective of the EA. This process is based on top-down modeling paradigm in which more concrete models are created from abstract models. First of all, business list, business purpose, vision, and business boundaries are described at the contextual layer. The value chains of the VE are modeled as CIM at the conceptual layer. Then, the detailed business components and the detailed business processes composing CIM are modeled as PIM at the logical layer.

As the VE is considered as a set of value chains which is made up of the collaborative business processes, it is natural that the enterprise models can be modeled with the process-centric approach. A generalized and process-centric representation for business partners was presented and implemented in our previous research [17]. As our CIM/PIM modeling phase is based on the modeling philosophy of the previous research, CIM and PIM for the enterprise models are regarded as the business scenarios and the collaborative business processes, respectively, as illustrated in figure 7.

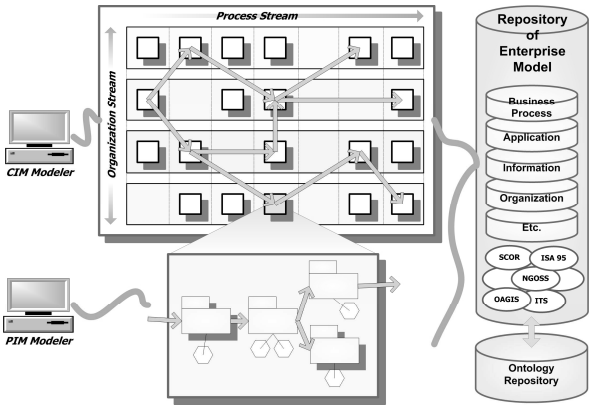


Figure 7. Modeling CIM and PIM

In the CIM modeling, we select some functional areas and organizations of each business partner using the function-organization matrix that describes the process stream and the organization stream of the collaborative enterprise. With respect to the process stream, units of enterprise activities need to be logically and temporarily ordered to realize the products of an acceptable quality at the right place and time. For the organization stream, organization, human and technical resources are systematically and repetitively assigned for the enterprise activities. The business scenarios are modeled through the meta-models for CIM.

In the PIM modeling, we perform the top-down analysis to reveal primarily the business processes and other business components according to the business scenario. The revealed business processes, other business components, and their relationships are systematically modeled through the meta-models for PIM.

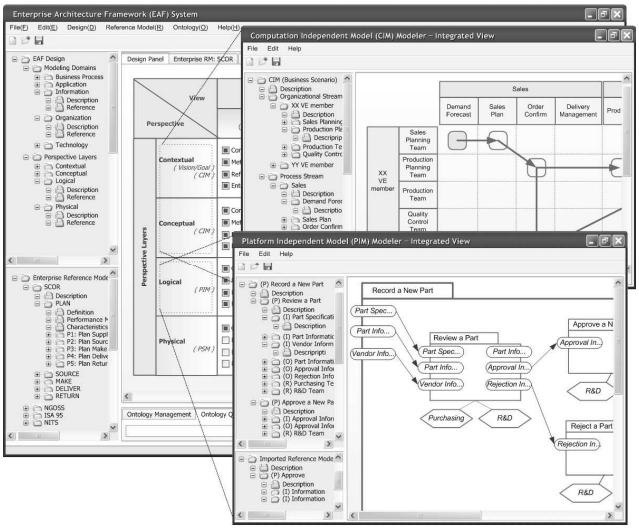


Figure 8. CIM Modeler and PIM Modeler

Figure 8 shows *CIM/PIM Modelers* for the CIM/PIM modeling phase and designed CIM and PIM examples. The phase of modeling CIM and PIM also makes progress on the EA. The designed enterprise models such as CIM and PIM are packed in the EA.

3.4. Deployment Phase

The designed enterprise models of the VE are deployed into PSM for actual execution so as to be suitable for specific technology platform.

In this paper, web service is considered as the best solution for PSM. Web service is a form of the SOA which is intended to enable developers to create service components that can be assembled and deployed in a distributed environment. Therefore, web service can be an ideal candidate for integrating enterprise application and setting up open and loosely-coupled information platform for the VE.

There are a set of key technologies and standards for web service such as Business Process Execution Language for Web Services (BPEL4WS) and Web Service Description Language (WSDL). They can be used for describing the web service models implementing the business processes of the VE.

In this paper, it is assumed that the deployment phase can make progress through the meta-model mapping between the meta-model of PIM and the meta-models of BPEL4WS and WSDL for transforming PIM to PSM, as shown in figure 9.

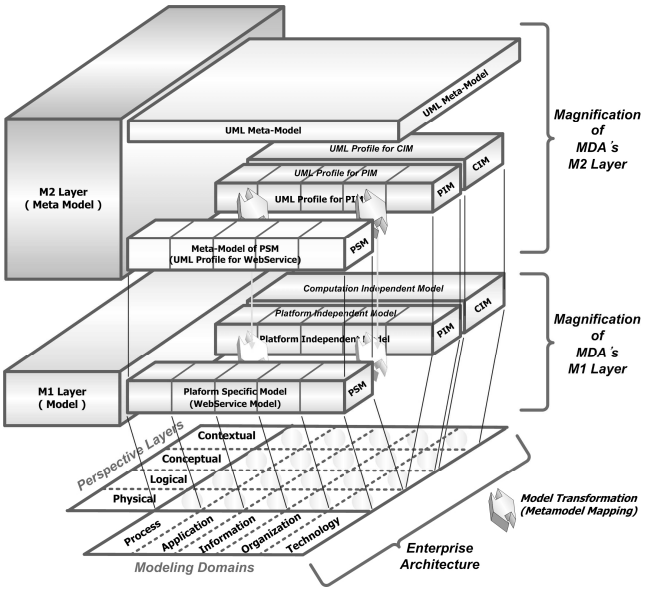


Figure 9. Model transformation in deployment phase

In other words, the meta-models of BPEL4WS and WSDL are developed as UML profiles and they are mapped into or from the meta-model for PIM. This meta-model mapping between meta-models at M2(meta-model) layer enable to transform PIM into PSM based on web service at M1(model) layer. The transformed PSM is expected to be possibly executed on some web servers such as Axis, WebSphere, or WebLogic.

4. Conclusions

The VE based on the process-centric loosely-coupled integration has become a key factor to survive under the competitive business environment. This research is originally motivated by the need for a systematic framework to contribute to configuration of the VE.

The proposed framework, named Enterprise Architecture Framework based on MDA, harmonizes several previous approaches which are dealt in different way on diverse domains and are developed independently to address its own purpose. Therefore, this framework can not only take individual advantages of each approach, but also produce integrated synergy effects.

This framework can be used for business managers or business domain experts to design the VE models in an elegant manner. This framework supports the configuration of the VE through the 4 modeling phases as follows: 1) EA design phase, 2) Meta-model design phase, 3) CIM/PIM modeling phase, and 4) Deployment phase through web service.

Although the framework is outlined in this research, rigorous further research for enriching this framework is currently underway as yet. The researches on the issue of the implementing systems are also undergoing and it is expected to get some more results soon. And a model transformation mechanism based on ontology in the deployment phase is being developed currently.

Acknowledgement

This work is partially supported by grant No. R01-2003-000-10171-0 and No. R05-2003-000-10080-0 from the Basic Research Program of the Korea Science & Engineering Foundation.

References

[1] CSC, "The Emergence of Business Process Management version 1.0", *A Report by CSC's Research Services*, 2002

[2] H. Smith and P. Fingar, *Business Process Management: The Third Wave*, Meghan-Kiffer Press, 2003

[3] H. Gou, B. Huang, W. Liu and X. Li, "A Framework for Virtual Enterprise Operation Management", *Computers in Industry*, 2003, 50, 333-352.

[4] P. Harmon, "Developing an Enterprise Architecture", *Business Process Trends: Whitepaper*, 2003.
[5] W. Barnett, M. J. Presley and D. Liles, "An Architecture for the Virtual Enterprise", *IEEE International Conference on Systems, Man, and Cybernetics*, 1994, San Antonio.
[6] J. A. Zachman, "A Framework for information systems architecture", *IBM Systems Journal*, 1987, 26(3).
[7] P. Harmon, "Enterprise Architectures", *Business Process Trends: Newsletter*, 2004, 2(1).
[8] CIO Council, "A Practical Guide to Federal Enterprise Architecture version 1.0", 2001, http://www.cio.gov
[9] DoDAF Working Group, "DoD Architecture Framework version 1.0 - Volume I: Definition and Guidelines", 2003
[10] IFIP-IFAC Task Force, "GERAM: Generalised Enterprise Reference Architecture and Methodology version 1.6.3", 1999, http://www.cit.gu.edu.au/~bernus/taskforce /geram/versions/geram1-6-3/GERAMv1.6.3.pdf
[11] P. Harmon, "Second Generation Business Process Methodologies", *Business Process Trends: Newsletter*, 2003, 1(5).
[12] Supply Chain Council, "Supply Chain Operations Reference Model – SCOR version 6.0", Supply Chain Council, Inc., 2003
[13] TeleManagement Forum, "Enhanced Telecom Operations Map (eTOM): The Business Process Framework version 3.0", GB921, 2002
[14] TeleManagement Forum, "System Integration Map version 2.0", GB914, 2002
[15] The Instrumentation, Systems, and Automation Society, "Enterprise-Control System Integration - Part 3: Models of Manufacturing Operations Management", ISA Draft 95.00.03, 2004
[16] J. Miller and J. Mukerji, "MDA Guide version 1.0.1", 2003, http://www.omg.org/docs/omg/03-06-01.pdf
[17] C.-H. Kim, Y.-J. Son, T.-Y. Kim, K. Kim and K. Baik, "A Modeling Approach for Designing a Value Chain of Virtual Enterprise", *International Journal of Advanced Manufacturing Technology*, Accepted and To be pressed in 2005 (Online Published: http://dx.doi.org/10.1007/s00170-004-2445-4).
[18] Object Management Group, "UML 2.0 Infrastructure Specification: Final Adopted Specification", 2003, http://www.omg.org/docs/ptc/03-09-15.pdf
[19] Object Management Group, "UML 2.0 Superstructure Specification: Final Adopted Specification", 2003, http://www.omg.org/docs/ptc/03-08-02.pdf
[20] Object Management Group, "UML Profile for Enterprise Collaboration Architecture Specification version 1.0", 2004, http://www.omg.org/docs/formal/04-02-05.pdf
[21] D. S. Frankel et al, "The Zachman Framework and the OMG's Model Driven Architecture", *Business Process Trends: Whitepaper*, 2003
[22] MetaCase, "Domain-Specific Modelling", *Application Development Advisor*, 2005, http://www.appdevadvisor. co.uk/express/vendor/domain.html

From high level business rules to an implementation on an event-based platform to integrate applications

Rubby Casallas, Catalina Acero, Nicolás López
{rcasalla, cata-ace, ni-lopez}@uniandes.edu.co
University of Los Andes, Colombia

***Abstract*—In this paper, we show how to build an implementation from a high level description of business rules through successive model transformations. The implementation of business rules can involve the integration of several heterogeneous applications. The key element of our proposal is the definition of a profile (EAI-Rules profile) whose objective is to define a vocabulary to model the concepts needed to integrate business activities and applications. The profile is used to annotate the models and, due to its well-defined semantics, we can assist transformations that lead towards a platform specific model that is ready to be executed. The specific platform is an event-based platform called Eleggua.**

Index Terms — Business Rules, EAI, MDA, Model Transformation.

INTRODUCTION

The accelerated pace at which Internet centered technologies have developed and the growing complexity of the nature of business have strengthened the need for organizations to clearly define business rules. These business rules should be transversal to the different processes in order to handle organizational knowledge in an integrated and coherent way. This need for integration at a process level directly influences the business application level that gives support to a process. Nowadays, business processes include the interaction of various applications, this situation raises the need to achieve an integration that has as objectives enabling fluid processes throughout the organization and providing a complete vision of each process.

The definition of business rules associated to the integration of business applications is a complex task. Some of the challenges are: (1) to achieve a shared knowledge of business domain concepts common to various applications; (2) to clearly identify the exchange of information between applications; taking into consideration that each application handles different data formats; (3) to precisely define the behavior and restrictions imposed by each rule; and (4) to validate the rules with the users who know the process in a language that is easily understandable for them.

The OMG has defined the UML profile for EAI [16] in order to give solutions to the challenges and to the needs generated to solve problems related to the integration of applications. The objective of the profile is "to define and publish a metadata interchange standard for information about accessing application interfaces" [16]. Its purpose is easing the tasks involved in the integration of applications.

The profile aims to define semantics, responsibilities, and restrictions of the elements that can make part of an integration infrastructure. Nevertheless, it still has several issues that make it hard to use on integration projects. The profile has a vague definition of some of its elements, their restrictions, lifecycles, and use. This makes its understanding and application a difficult task. In the end, what usually happens is that each project develops an implementation of an integration infrastructure that is not compliant with any standard.

We have defined a new profile, called *EAI-Rules* profile, with the purpose of filling some of the gaps and issues detected in the EAI profile. The main objective of the profile is to define a vocabulary (i.e. ontology) to model the concepts needed to integrate business activities and applications. Using the EAI-Rules profile, we have defined the following approach that aims to give some answers to the challenges presented above. (1) Our profile introduces a vocabulary that enables the definition of integration business rules at a high level of abstraction (i.e., to the Computation Independent Model [11]) (2) We assist the transformation of the model into a PIM (Platform Independent Model) that has as metamodel the EAI-Rules profile. (3) We propose a transformation scheme to build an implementation of the PIM by merging it with a PSM (Platform Specific Model), which in this case is aligned with the implementation model of an infrastructure called Eleggua [13].

The work presented in this paper is part of a project, developed by the Software Construction Group, at the University of Los Andes. The global project[1] has as main objective the definition and implementation of an infrastructure for application integration and cooperation to support Global Software Development [14].

This document is organized in the following way: Part 1 presents the main characteristics and issues of the UML for EAI Profile. Part 2 introduces our EAI-Rules profile. The purpose of Part 3 is to show how business rules at a high level are expressed using an activity model annotated with the elements of our profile. Part 4 establishes how the activity model can be transform into a model (PIM) based on our profile. Part 5 presents the transformation rules from the PIM to a PSM. Part 6 introduces some related works and compares the approaches. We finished the paper with some conclusions and future work.

EAI PROFILE

The EAI profile is a UML profile defined by the OMG (Object Management Group) [16]. The profile provides a standard mechanism to define an application integration model using UML as language.

[1] The project is supported and partially financed by the "Instituto Colombiano para el Desarrollo de la Ciencia y la Tecnología Francisco José de Caldas"-COLCIENCIAS. Colombia.

Scenarios of Use

The scope of the profile contemplates three scenarios of use. Scenario 1 considers the integration of applications through connectivity. Applications share a common architecture and data model for communication. The integration can be synchronic or asynchronic, and it must be possible to model service requests, responses and notifications.
Scenario 2 considers integration of applications that need to share information; it describes how application share information using pub/sub business events as communication. In this scenario applications share business events and the model of the process at a conceptual level. Nevertheless, at a low concrete level, data models can be different. One of the objectives of this scenario is that it should be simple to add new participants and services to an existing infrastructure.
Scenario 3 considers integration of applications through collaboration processes; it describes Business to Business (B2B) integration of businesses. In this context, integrated applications can be located on different organizations and have completely different business domains.

The Metamodel and the profile

The EAI working group defined a metamodel; this metamodel of integration is a specialization of the FCM (Flow Composition Model) of the EDOC profile (Enterprise Distributed Object Computing [4]. The EAI metamodel characterizes aspects related to connectivity, composition and behavior of the integration elements. At a general level, this defines the syntax and semantics of elements such as EAILink, EAITerminal, EAIMessage, EAISource, EAISink, EAIAdapter, EAIFilter, etc.
Based on the metamodel, the EAI working group has defined its UML profile. The profile consists of two main components: activity model and collaboration model. The first enables modeling of control and data flow between applications involved in the integration. The second provides more detail about the semantics of collaboration. In particular, this describes the exchange of messages between applications.

Discussion

The EAI profile is a potential tool to define, in a standard way, any integration model. However, some open issues make difficult the understanding and practical use of it in a real context [3] [5] [7] [10] [12].
Some of the elements presented in the profile are incomplete. Aspects related to their semantics, lifecycle and constrains do not have enough detail to enable a transformation toward an implementation.
Additionally, the relationship between the two components constituting the profile is not clear. The profile is missing a mapping that correlates the concepts shown in the activity model and those in the collaboration model. Software Architects have to wait until implementation to make important decisions because of these gaps. Moreover, currently no industrial provider has completely implemented the profile.

To fill some of the gaps and issues detected in the UML for EAI profile, we have defined the EAI-Rules profile and present it briefly in the next section.

EAI-RULES PROFILE

EAI-Rules profile is a vocabulary that contains elements to model application integration. The scope of this profile is scenario 1, integration of applications through connectivity, and scenario 2, integration of applications through shared information. The profile is expressed using UML to define stereotypes and OCL [17] to define the restrictions on the elements.

In this section, we present three new concepts missing in the EAI profile. Other stereotypes are introduced later as needed. The three concepts are: 1) activities of interest that trigger events of integration 2) flow of information between activities, and 3) actions executed as response to events. The profile contains the semantics, responsibilities, and, restrictions of each element and the relationship with each other in the profile.

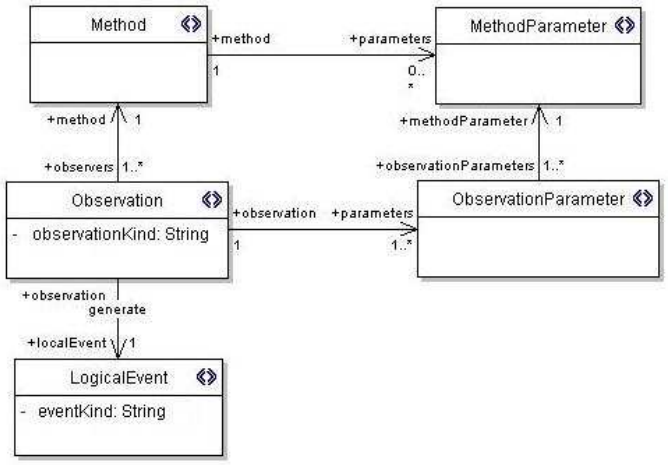


Fig. 1
MODEL OF OBSERVATIONS

Activities of interest

To model activities of interest that trigger events of integration, we have defined the concept of *Observation*. An observation has the responsibility of intercepting the execution of a method in an application and generating a logical event of interest to other applications. Figure 1 shows a UML diagram of the main elements involved in an Observation. The Observation observes the execution of a method, there is a relationship between the parameters of the observation and those of the method, and an association between the observation and the generated logical event.

Flow of Information

To model the flow of information between activities, the profile defines the concept of *Logical Event*. A logical event is defined by its type (EventType in figure 2) and its parameters

(EventParameter in figure 2). The event type of the logical event, represents a domain concept common to a group of applications. Applications interested on receiving notifications of a particular event type, have to subscribe to it.

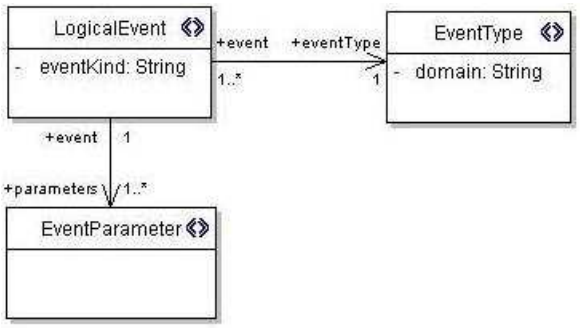


Fig. 2
MODEL OF LOGICAL EVENTS

Other stereotypes related to events are presented later to annotate the business rules.

Actions

To model the actions executed as a response to an event, the profile defines the *ECA Rule* concept. An event type, a condition or filter, and a set of actions define an ECA rule. A notification of a logical event triggers the execution of a rule whose type is the same as the one defined in the rule. Once the rule is triggered, if its condition is validated, the set of actions is executed.

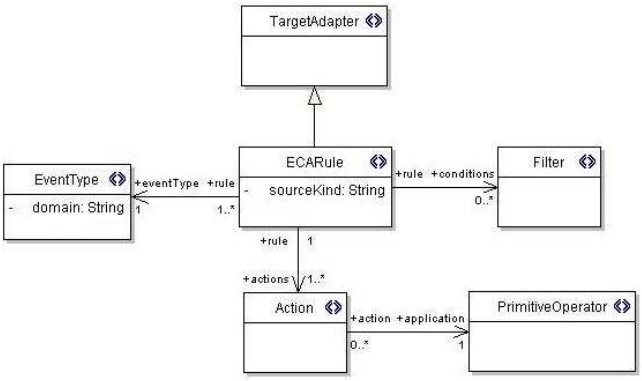


Fig. 3
MODEL OF ECA RULES

These actions can generate new logical events or invoke external application's services. Figure 3 presents the UML representation of the main elements involved in the definition of an ECA Rule.

BUSINESS RULES ANNOTATED WITH THE EAI-RULES PROFILE

We now present a simple example scenario to illustrate the use of the profile. The example scenario takes place in the context of the business processes of a software development house. This organization has two fundamental business policies: (1) Planning and tracking all the activities, and (2) Cost assessment based on employee time logs registered for work activities.

These policies derive various business rules; we have selected two of these in order to illustrate our approach:
R1: For each defect detected during the execution of a test plan, an activity for correction has to be created and assigned to the responsible user.
R2: The total time spent performing the defect correction task has to be recorded for quality metrics purposes.
The rules above are expressed in plain English. The ultimate objective is to transform these rules to something executable automatically. The following sections present the steps to pursue this result. Each step consists of assisted transformations based on the models.

CIM transformation

We have defined two assisted transformations to express the business rules in a particular CIM: (1) Transformation from plain English to an activity diagram that specifies the tools that support the process and shows the exchange of information between activities. (2) Annotations on the elements of the diagram (activities, transitions, and events) with the stereotypes defined in the EAI-Rules profile.
Figure 4 summarizes the result of both transformations on rules R1 and R2. The example scenario includes the interaction of two applications: *Hammurabi*, which is used to create software-testing plans, registration of results of test executions and creation of defect reports; and *Cronos*, which is used to plan activities and register time logs.
The main points of interaction that need integration for these applications are: (1) when a defect correction is assigned to a user in *Hammurabi* (Fig. 4 activity 1), automatically, *Cronos* is notified of this event and creates a new task for the developer in charge (Fig. 4 activity 2). Afterwards, the developer in charge has to register the time spent performing the activity (Fig. 4 activity 3). (2) Once the developer finishes and closes the task, (Fig. 4 activity 4) total time spent is sent to *Hammurabi* for generation of reports on defect corrections.

Annotations using the profile

The elements on the activity diagram have been annotated with stereotypes belonging to the EAI-Rules profile (Figure 4). These stereotypes are:

- **Observation:** annotates the activity *Assign defect correction*. It indicates that the execution of the *assignDefect* method is observed and as a consequence, a logical event is generated.
- **LogicalEvent:** annotates an element, of the activity diagram, Signal Sending or Signal Receipt. Signal Sending indicates the logical event generated by an observation. In fig-

ure 4, *defectAssign* is the event generated by *Hammurabi* (produce stereotype) and received by *Cronos (*consume stereotype*)*. The figure shows also the *LogicalEvent* annotation on the Signal Sending and Receipt.

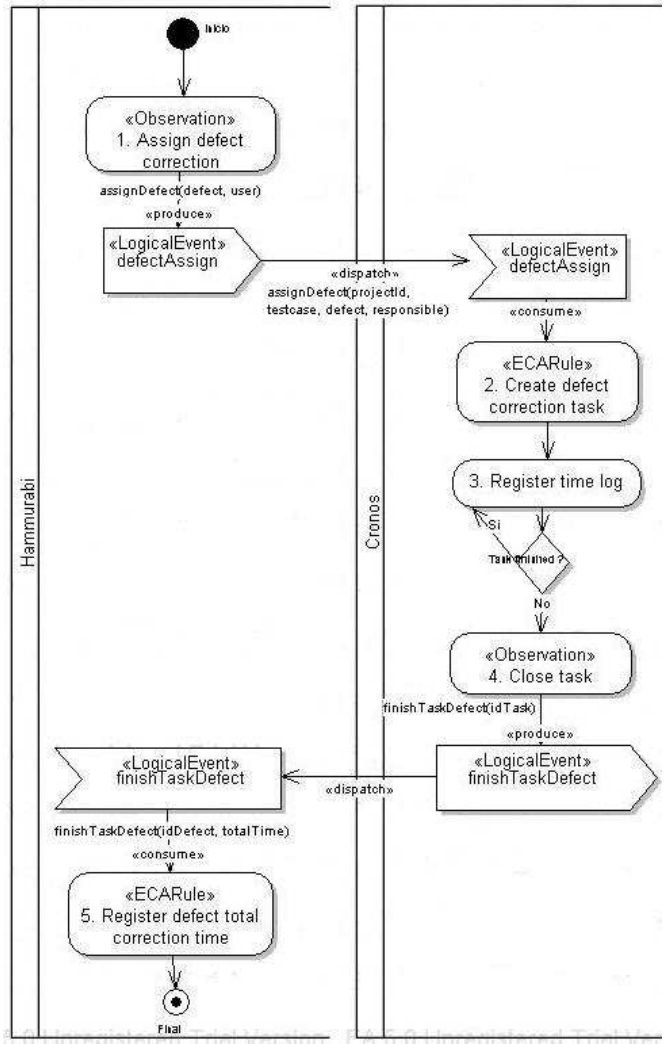


Fig. 4
ACTIVITY DIAGRAM

a task to correct the defect.

The next section presents how to perform a transformation to a PIM from the information provided in the activity model.

TRANSFORMING BUSINESS RULES TO A PIM

We use the example scenario to show the transformation. This section presents the transformation required to model the interaction between *Hammurabi* and *Cronos* when a defect correction is assigned in *Hammurabi*.

Transforming an observation

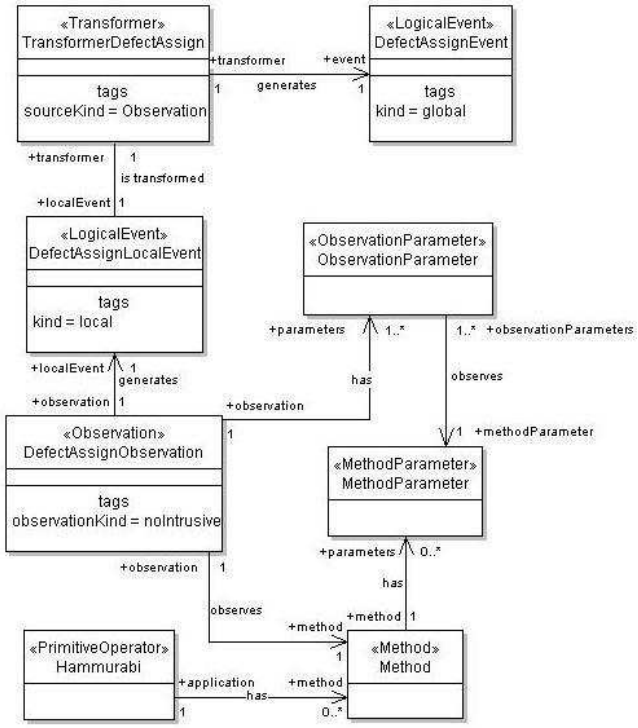


Fig. 5
LOGICAL EVENT PRODUCTION

- **Dispatch**: indicates that a logical event generated by an application is dispatched to applications interested in events of this type. In this dispatching process, the event can go through a transformation according to the business rules. The association between the SignalSending and SignalReceipt is marked with the stereotype dispatch because it represents the process of dispatching of the event from the producer application to the event consumers. In this case, before the dispatching process, the logical event can be transformed and new parameters can be added to it.
- **ECARule**: indicates activities that are triggered as reaction to a previously generated logical event. In figure 4, the activity *Create defect correction task* has an *ECARule* annotation. This means that when the event *defectAssign* is consumed, the rule is triggered and it creates automatically, to the developer in charge,

Figure 5 shows the class diagram corresponding to the observation and generation of a logical event to assign defect correction. The stereotype *PrimitiveOperator* identifies *Hammurabi*, this element is used to represent system applications. The application has several methods some of them are of interest for the integration model. In those cases, an Observation element has to be used to model the interception of the methods.

The transformation is achieved in the following way: (1) the swimlane (Fig. 4) represents activities that are realized in *Hammurabi*. This external application is represented in Fig. 5 as a *PrimitiveOperator*. (2) *Assign defect correction* activity (Fig. 4) is represented as an Observation element (*DefectAssignObservation* in Fig. 5) that is intercepting the execution of the *assignDefect* method. (3) Arguments for the

assignDefect(defect,user) method are used to complete the observation parameters.

Transforming a logical event

The execution of the method of interest, in this case *assignDefect*, triggers the observation whose execution generates a *LogicalEvent* of local type. The Transformer transforms this event into a *LogicalEvent* that might have more information, data transformations or filtered information.
In figure 4, the link that joins the SignalSending and SignalReceipt elements is of type *dispatch*, this is represented in figure 5 by the class *TransformerDefectAssign*, which adds information to the event and generates a new *LogicalEvent*. In the activity diagram, the link that joins activity 1 with the logical event *defectAssign* has the *produce* stereotype, this indicates that the logical event must be published; this is represented through the use of the *PublicationOperator* (Fig. 6).
Once the transformation generates a logical event, it is sent to a *PublicationOperator* element. This element notifies all interested rules of the occurrence of the event. To perform the notification, the *PublicationOperator* relies on the *SubscriptionTable* element. This shared resource stores information on subscriptions to the event types in the system (Fig. 6).

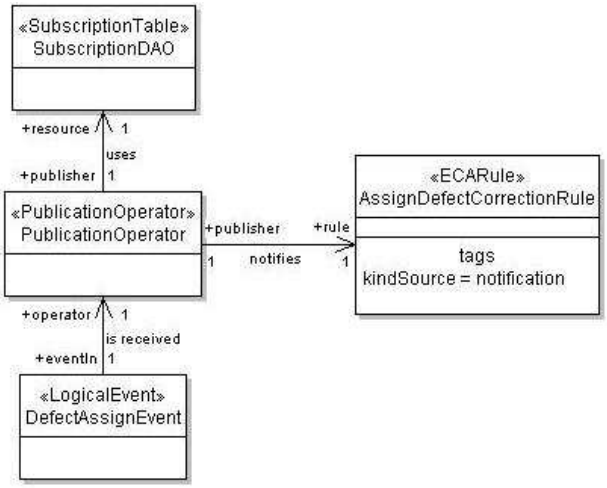


Fig. 6
LOGICAL EVENT DISPATCH

Transforming a ECA Rule

Through the *PublicationOperator* element, the ECARule, subcribed to the event type, is notified of the occurrence of the event. Once the *DefectAssignAction* rule receives the event, it evaluates its conditions and executes the specified actions. In our example, the action creates a task for the correction of the defect.
Figure 7 shows the result of the transformation from the activity diagram. The steps were: (1) the link marked with the stereotype *consume* is represented in the diagram using the *PublicationOperator* and *ECARule* elements. (2) A condition in the ECARule is included by default to validate that the parameters from the event match the ones needed by the rule. In this case, the condition checks that the event has as parameters the identification for the project, test case, the defect and the responsible user. (3) Finally, the application, modified by the action, is represented by a *PrimitiveOperator*.

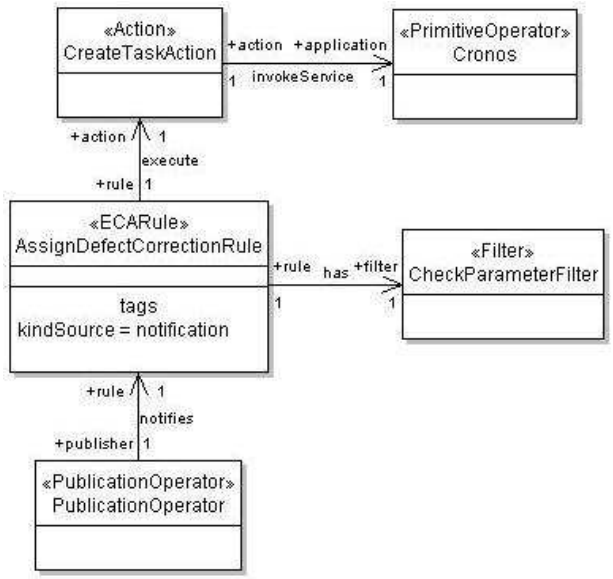


Fig. 7
LOGICAL EVENT CONSUME

ELEGGUA PLATFORM

Figure 8 presents the main components of the Eleggua platform. These are: in the center, a distributed Event Notification System (ENS) and around it, the Application Representatives (AR), one for each external application involved in the integration. The ENS provides event notification services and the ARs mediate the communication of applications with the ENS.

Each Application Representative executes and manages Observations and ECA Rules. The distributed ENS manages event types and subscriptions and offers services for event dispatching and notification.

Figure 9 presents the components and relationships of the infrastructure that implements the components in Fig. 8: the DEM (Distributed Event Middleware) and the CP (Cooperation Proxy) that implement the ENS and Application Representative respectively.

The DEM is a distributed component that offers the basic functionality of an ENS: subscription to event types and event notification. The CP implements the application representative by offering services for registration and processing of ECA Rules and Observations.
The chosen implementation is based on Enterprise Java Beans [15]. The *ECARuleProcessor* (Fig. 9) is a session bean that processes logical events generated by the observation processor (local events) or notified by the DEM (global events).
Figure 10 presents the *ObservationProcessor* and other main

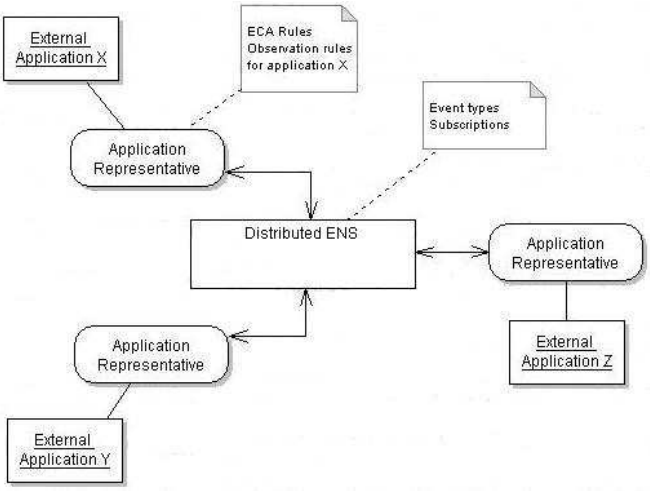


Fig. 8
ELEGGUA PLATFORM

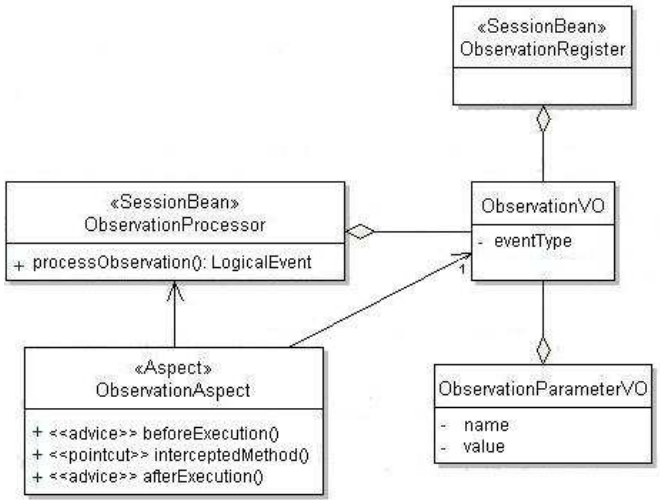


Fig. 10
OBSERVATION PROCESSOR ELEMENTS

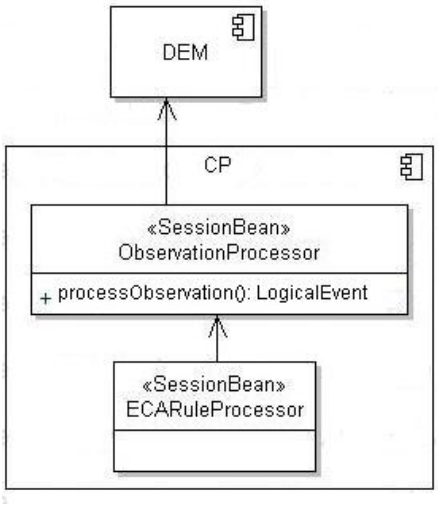


Fig. 9
COMPONENTS OF ELEGGUA

elements of the specific implementation for observations. The *ObservationProcessor* offers services to the aspect class *ObservationAspect*. This aspect class intercepts method execution and creates an *ObservationVO* object with its *ObservationParameterVO* objects for processing. The *ObservationRegister* bean offers services for registration of new observations. Aspect service interception uses AspectJ technology [1].

During execution, the aspect class that implements *ObservationAspect* intercepts the execution of the method and creates an *ObservationVO* object passed to the *ObservationProcessor*.
Figure 11 presents the *ECARuleProcessor* and other main elements of the specific implementation for ECA Rules. The *ECARuleProcessor* receives logical events for processing and executes them using a *RuleExecutorClass* for an ECA Rule that matches the event type. The method *checkCondition* is used to evaluate the condition of an ECA Rule; the *execute* method implements the actions of the rule.

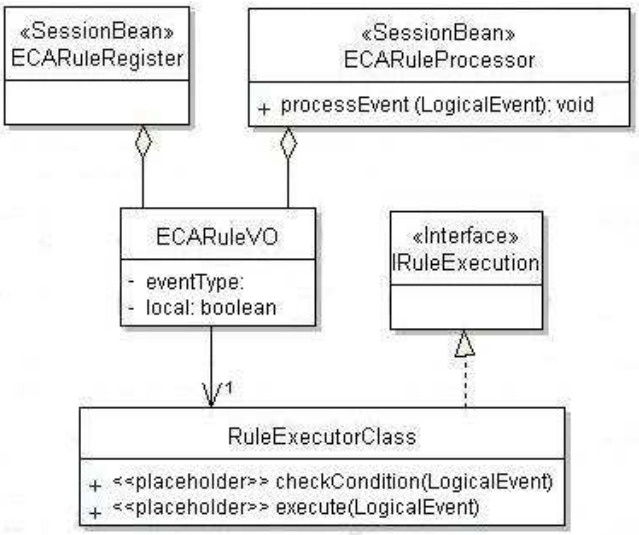


Fig. 11
ECA RULE PROCESSOR ELEMENTS

TRANSFORMING THE PIM TO A PSM

We now describe the main elements of a transformation from the PIM to a PSM based on Eleggua. For reasons of space, we only give a brief description of the transformation of the main elements described in the PIM: observations, events, and ECA rules.

Transforming an Observation

An aspect class that extends *ObservationAspect* and implements the point cuts and advices represents each Observation element from the PIM. This aspect intercepts the execution of a given method and passes the observation to the *ECARuleProcessor*.
During execution, the aspect class creates an *ObservationVO* with the *ObservationParameters* and *MethodParameters* modeled in the PIM related to the observation element.
An *ECARuleVO* represents the transformation element referenced by an Observation. The *ECARuleVO* has it local attribute set to true. This local rule executes the transformation and notification of a Logical Event instance to the DEM.

Transforming a logical event

The *LocalLogicalEvent* modeled in the PIM referenced by the Observation is modeled during execution as a *LogicalEventVO* produced by the *ObservationProcessor* session bean.
A *LogicalEventVO* generated by the ECA Rule that represents a transformation element in the PIM models the logical event referenced by the transformation.
The DEM manages and stores event types and subscriptions. The *ObservationProcessor* passes its produced events directly to the *EventProcessor*. A local *ECARuleVO* represents each transformation element from the PIM in the PSM.

Transforming an ECA rule

A global *ECARuleVO* represents each ECA Rule element with its respective filter element from the PIM. Additionally, the transformation includes creating a class that extends the *RuleExecutorClass*, this class has placeholders in the *checkCondition( )* and *execute( )* methods. Placeholders indicate places in the code that the developer must complete. These methods implement the filter element and ECA Rule element from the PIM.
The transformation also assists the regeneration of the external application code to include AspectJ interception of methods for observations. The transformation generates a build file with placeholders that recompiles and redeploys applications.

Comparison with other works

The BRMS (Business Rule Management System) deals with the problem of maintaining a system due to changes in the business rules. Usually, a business rule is set in the code of the several components affected. A BRMS aims to overcome this problem by "separating the definition of policy from implementation and code details" [8].
There are several BRMS in the market, such as ILOG JRules [9], HaleyRules [6], Blaze Advisor [2] and some more pointed mentioned in [8].
The difference between our proposal and those BRMSs resides in the scope and in the approach. Concerning the scope, our work specifically addresses the problem of defining business rules, which involve integration of applications. Our assumption is that the execution of a complex business process involves activities supported by different (distributed) applications.
In relation to the approach, ours is an approach based on MDA. Even if we share with the other BRMS systems the expression of business rules in a high-level language close to the user language; the transformation towards an implementation is different. The advantage MDA gives lies on the transformation functions, which can produce different implementations from the same model. In JRules the transformation is done always to the (ILOG Rule Language), which is the language the rule engine can execute.
Furthermore, to define business rules, in our case, the user has to know the services of interest provided by the applications, and be familiar with the EAI-Rules profile. The user is unaware about any specific implementation; nevertheless, using JRules, the user has to know the BOM (Business Object Model), which is quite close to the java classes in the implementation.
A current disadvantage of our work has in comparison to the market is that most of the systems have powerful tools to define the rules and to monitor and administrate their execution; we are not there yet. As we explain in the next section this is part of our future work.

Conclusions and future work

In this paper, we have presented an approach to transform high-level business rules to a specific implementation platform. The approach uses the EAI-Rules profile, based on the EAI profile, provides a vocabulary that contains elements to model application integration.
We also presented an example scenario to show how the profile can be used to describe business rules. The example presents a transformation from a CIM to a PIM annotated with the profile using activity diagrams and the stereotypes defined. Finally, we introduced a platform (Eleggua) that implements the profile and a brief explanation of transformation from the PIM to a PSM based on Eleggua.
The presented profile and transformations aim at reducing the complexity involved in the definition of business rules. Nevertheless, the transformations presented are assisted, but are not completely automatic. A lot of work is left to be done in developing tools that reduce the work
involved in the transformation process and ease the administration of the rules.
On the other hand, the validation of a PIM model is an area that still needs further work. We are currently designing tools that help model and simulate a business process PIM before a transforming to any PSM using executable UML.

References

[1] AspectJ team. http://eclipse.org/aspectj/. Last visited on 2005-01-23.

[2] Blaze Advisor. Business Rules – Fair Isaac Corporation. http://www.fairisaac.com/Fairisaac/Solutions/Enterprise+Decision+Management/Business+rules/. Last visited on 2005-01-08.

[3] DSTC, Hitachi, Ltd., IBM Corporation, Oracle Corporation, Rational Corporation, Unisys Corporation. UML Profile

and Interchange Models for Enterprise Application Integration (EAI) Specification Issue Reporting. 2001

[4] Flow Composition Model (FCM) Specification. www.omg.org. February 2004.

[5] Giorgio Bruno, Marco Torchiano, Rakesh Agarwal. UML Enterprise Instance Models. ACM Software Engineering Notes, 28(2), March , 2003.

[6] Haley Systems Business Rules Management Systems. Natural Languages – Business Rules. www.haley.com. Last visited on 2005-01-08.

[7] Hans-Peter Hoidn, Dave Smith. PricewaterhouseCoopers. UML for EAI – Report to the ADTF. OMG TC Meeting. November, 2001.

[8] Ian Graham. Trireme. Service Oriented Business Rules Management Systems. Version 2. www.trireme.com/Service_Oriented_ Business_Rules_Management_Systems.htm June, 2005.

[9] ILog JRules 4.6. White paper. www.ilog.com. April 2004.

[10] John Knapman. UML for EAI. UML Profile and Interchange Models for Enterprise Application Integration (EAI) Revised Submission.. neptune.irit.fr/Biblio/01-11-21.pdf. 2001.

[11] MDA Guide. Version 1.0.1. www.omg.org. 2003

[12] OMG. Documents & Specifications, Report a Bug/Issue. www.omg.org/issues/issue4853.txt. Last visited on 2005-07-01.

[13] Rubby Casallas, Nicolás López, Darío Correal. An Event Infrastructure for Application Cooperation. To appear in Proceedings of Component-Oriented Enterprise Applications (COEA 2005). Erfurt, Alemania. September, 2005.

[14] Rubby Casallas, Darío Correal, Nicolás López. Mejoramiento del proceso de pruebas en un contexto de desarrollo de software globalizado. To appear in Proceeding of the XXXI Latin-American Conference CLEI. Cali, Colombia. Octubre. 2005.

[15] Sun Microsystems. Enterprise JavaBeans Specification. Version 2.1. http://java.sun.com/products/ejb/docs.html. November, 2003.

[16] UML Profile and Interchange Models for Enterprise Application Integration (EAI) Specification. Version 1.0. www.omg.org. Mars 2004.

[17] UML Unified Modeling Language Specification.. www.omg.org. Version 1.5. March 2003.

Rule-based business process modeling and execution

Stijn Goedertier Jan Vanthienen

***Abstract*— A process model is called rule-based if the semantics of its case data and activity flows are expressed by means of rules. Rules have been recognized before as powerful representation forms that can potentially define the semantics of data and process resources. To date, however, there is no consensus on how to link the enforcement of rules, the manipulation of data and the execution of processes. Moreover, it is witnessed that complex data and process resource descriptions in the form of a number of constraints, deduction and reaction rules lack expressivity and comprehensibility. In this paper, we set out for using process and rule set metamodels to concisely represent the semantics of case data and activity flows in business process models. In addition we show how to generate a syntactically verified and semantically validated corpus of definite Horn clauses that can be used in the execution of the modeled process.**

I. Introduction

Software engineering aspires to avoid the duplication of resources. This is a fundamental principle which is based on the experience that systems with duplicated resources sooner or later run across a myriad of difficulties. As with data in the past and process descriptions at present, logic is gradually becoming the next resource to be managed outside individual applications. Through the separation of so called business rules from applications, it is hoped that changes in business logic will no longer result in an avalanche of required application updates, and will thus reduce the IT bottleneck when bringing about business policy changes. Separating logic from applications is the goal of *rule-based software engineering*, which has the potential of significantly improving the theory and practice of system design.

This separation, however, raises the problem of how to link the enforcement of business rules, the manipulation of data and the execution of processes. Several approaches to this problem are described in the literature, such as Dietrich's rule-based agents [1] and D'Hondt's approach which considers business rule enforcement as an aspect-oriented programming cross-cutting concern [2]. In this paper, business rule enforcement is situated at the level of business processes rather than at the level of individual applications. In particular, we propose a process and rule set ontology for *rule-based business process modeling* and an architecture for *rule-based business process execution*.

A. Existing process languages

Rule-based process modeling is different from existing process languages, because it allows to define both the flow-control and data perspective of business processes. Formal process languages like Petri nets [3] and π-calculus [4] predominantly focus on the flow-control perspective of business processes and validate sequence constraints only. Likewise, process execution languages like BPEL attach little attention to declarative formulation of case data semantics. In such process languages it might be possible to simulate aspects of data semantics using a process description. The latter, however, is often argued to overburden process descriptions. Moreover, the inclusion of data semantics in individual process models goes against the above mentioned principle to avoid duplication of resources. *Rule-based business process modeling*, in contrast, has the potential of aligning the semantics of both business vocabulary and business process descriptions in a natural and concise manner.

Katholieke Universiteit Leuven,
Department of Decision Sciences & Information Management
Naamsestraat 69 - 3000 Leuven - Belgium
stijn.goedertier@econ.kuleuven.be, jan.vanthienen@econ.kuleuven.be

B. Existing rule languages

The literature categorizes business rules in three basic types [5]: constraints and derivation rules, which define the semantics of data resources, and reaction rules, which define the semantics of process resources. Rules have been recognized before as powerful representation forms that can potentially define the semantics of data and process resources. Dietrich et al., for example, describe the behavior and knowledge of an artificial agent using a set of derivation and reaction rules [1]. In this paper we enhance the expressivity and comprehensibility of this approach, making it suitable for rule-based process modeling. Expressivity is enhanced, because we allow process rules with composite events [6] and long-running activities. Comprehensibility is enhanced because we consider rule sets rather than rules as the atomic unit of logic.

The remainder of this paper is structured as follows. First we give an outline of the architectural context of rule-based business process execution and situate business rules in enterprise models. Next we define a rule-based process metamodel that can be used for describing business processes as sets of process rules, which facilitates alignment with other types of business rules. To enhance comprehensibility, we define a generic rule set metamodel that is useful in representing different types of rule sets. Finally, we display excerpts of a process description and show as a proof-of-concept how to generate a corpus of definite Horn clauses from it that can be used in the execution of the modeled process.

II. The architectural context of rule-based process execution

Data, processes and logic are essential resources of any information system and can consequently be identified at any level of abstraction. For the purpose of rule-based process execution, it is useful to consider these resources at the highest level of a service-oriented enterprise-architecture stack [7]. Such an architecture stack, as displayed in figure 1, commonly consists of a number of layers. Applications and databases of one layer are concealed by the components and services of a higher layer. Services can be combined forming long-running business processes, which make up the highest layer.

Business process models are flexible descriptions of long-running interactions between business partners. Process in-

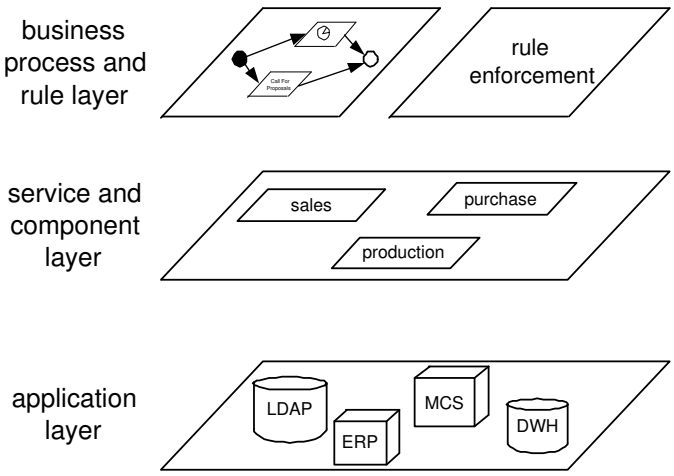


Fig. 1
ARCHITECTURAL CONTEXT

stances represent a sequence of activities that are triggered by business events and that represent the invocation of services. The execution of business processes is governed by a so called *process engine*. In the context of rule-based process execution, a process engine has the following components: a message handler, a persistence mechanism for case data, an inference engine and a mechanism to invoke services once their corresponding activities have been initialized.

In correspondence with the perception-reaction cycle described in [1] and the decomposition principle in reactive agent architectures in general, we decompose the processing of an event into a series of aspects, that can be addressed via a number of consecutive logic queries on an inference engine. Each query refers to rules, which are to be derived from an enterprise model.

0. **Derivation rules:** define facts that can be logically derived from other case data elements. Derivation rules are used throughout the entire event processing cycle.
1. **Authorization rules:** is the participant authorized to raise the particular event? If this is not the case, any further processing of the event message stops. Is the participant authorized to provided the data elements that are contained by the event message? The data elements that violate authorization rules are left out.
2. **Input validation rules:** do the data elements – if any – that accompany the event satisfy the business constraints, given the available case data elements? Only if this is the case, these data elements are incorporated in the case data of the process instance.
3. **Case data requirements:** are all data elements that are required at this or future decision points available? If this is not the case, these data elements should be collected from the proper process participants or underlying database and application layer. The advantage of this approach is that it no longer requires process participants to exchange messages of which the content is fixed a-priori.
4. **Process rules and activity preconditions:** Given the new event, and the events, data elements and activity states that make up the state of the particular process instance, which – if any – activities should be launched with which parameters? Are the preconditions, such as sequence constraints, of the activities to initialize satisfied?
5. **Notification rules:** in case an activity is launched, which participants should be notified and what data elements correspond to this notification?

III. RULE-BASED BUSINESS PROCESS MODELING

Enterprise models are abstractions of different aspects of an enterprise, typically with a purpose to understand and share the knowledge of how the enterprise is structured and how it operates [8]. In addition to the purpose of knowledge management, enterprise models can be a foundation for model-driven-architecture. In the context of rule-based business process modeling, three submodels of an enterprise model are particularly relevant, as displayed in figure 2.

- A *Business Vocabulary model* contains the knowledge artifacts of an enterprise model. It is often argued to express Vocabulary Models using fact-oriented ontology languages rather than object-oriented ontology languages [9]. At this level of abstraction, little matters how attributes resort under objects. In the context of rule-based modeling, fact-oriented models are particularly useful because they are closer to natural language and logic programming, than object-oriented models. For the purpose of this paper, we settle for a vocabulary language with only two constructs: unary predicates to indicate domain classes and binary predicates to indicate domain properties. These constructs correspond to the class and property constructs of semantic web ontology languages [10]. In this paper, however, we refrain from using any specific ontology language for representing business vocabulary.
- A *Business Process Model* consists of process descriptions that describe how the enterprise interacts with external process participants and which internal services should consequently be invoked. To enable rule-based process modeling, we devise a lightweight business process metamodel of which the constructs are outlined in the following section.
- Business rules models govern the dynamics of data and process resources. Consequently, Vocabulary and Process models are related to business rules models [8]. Vocabulary models are related to *Business Rule Models*, because business logic contains derivation rules and constraints that define or constrain predicates of the vocabulary model. Constraints, for instance, can define the (conditional) cardinalities of artifact relations. Derivation rules define predicates that can be logically derived from the available case data. Process models are related to *Business Rule Models* because business logic defines the state transitions of process instances. This is displayed in figure 2. Business rules are undoubtedly powerful representations, but as we will argue, rule sets rather than individual rules should be considered as the atomic unit of logic in enterprise models. To this purpose, we define a generic rule set metamodel in section V.

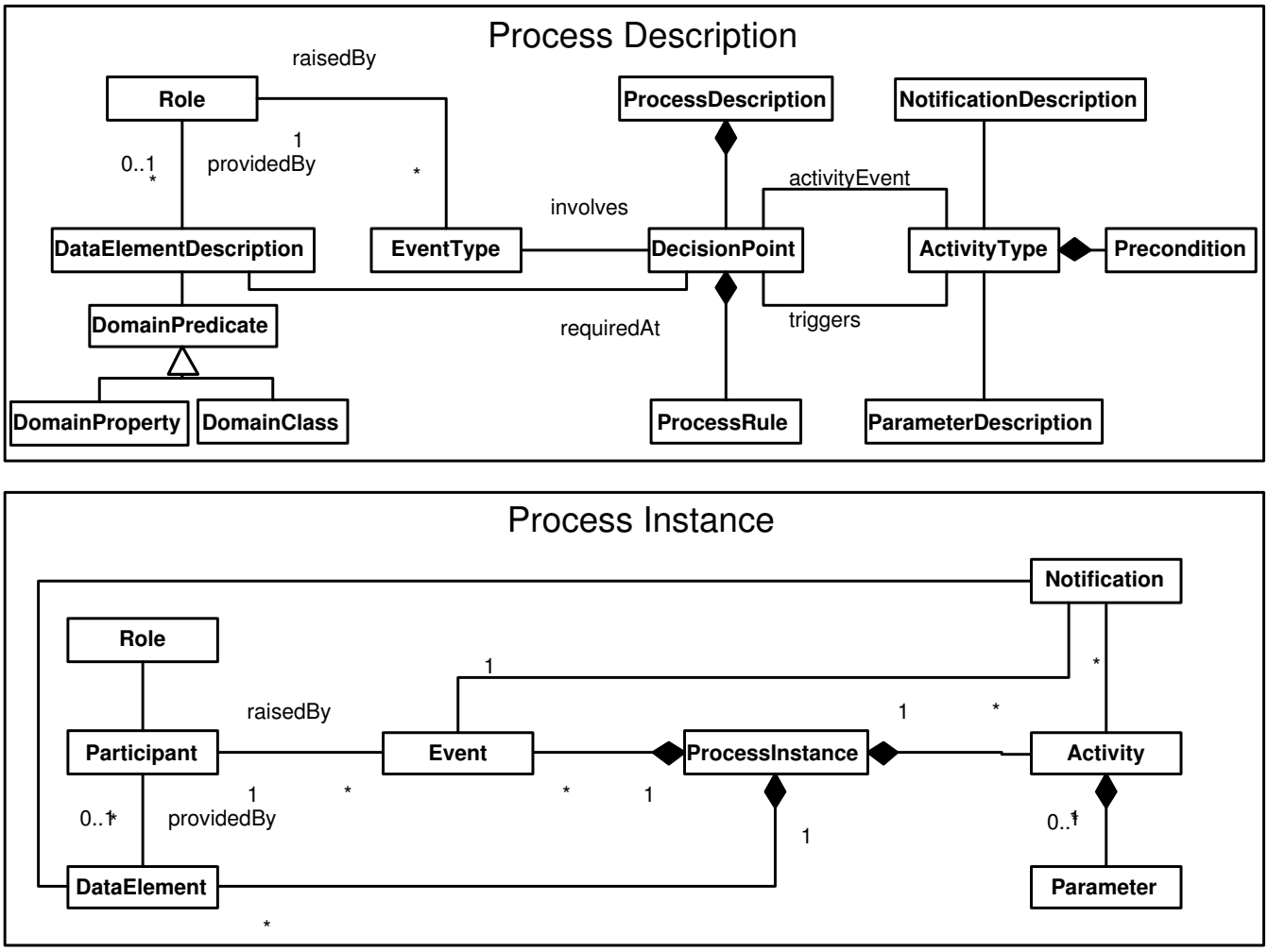


Fig. 3
A LIGHTWEIGHT PROCESS METAMODEL, AT DESCRIPTION AND INSTANCE LEVEL

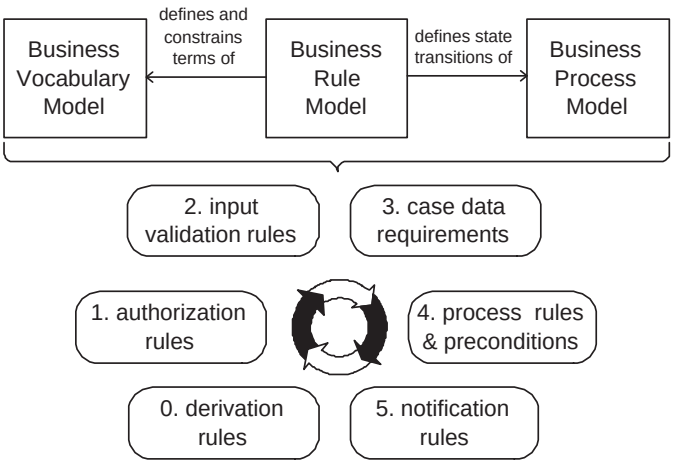


Fig. 2
BUSINESS RULES IN ENTERPRISE MODELS

IV. A RULE-BASED PROCESS METAMODEL

At this point it is not our intention to thoroughly describe the precise semantics of yet another process language. Instead, we specify a lightweight process ontology, that can be the foundation for a rule-based process language of arbitrary complexity. The main building block of this process ontology is a decision point, in which control flow can be described using a set of process rules. This rule-based nature of the control-flow description facilitates its integration with rule-based case data semantics. How advanced control-flow patterns such as cancelation and multiple instance patterns [11] fit into this rule-based framework is outside the scope of this paper.

A. A process instance metamodel

Figure 3 represents a MOF/UML metamodel of rule-based processes both at the level of process instances and process descriptions. Let us begin our description of the process ontology at the level of the *process instance*. In our view, a process instance's state is defined implicitly by its data elements, events and activity states. *Activities* are long-running, asynchronous invocations of services that are initiated by the process engine.

Depending on the required semantics of the rule-based process language, activities might have states such as 'initiated', 'executing', 'failed', 'canceled' and 'succeeded'. In the process language we use in our example, the states 'failed' or 'succeeded' are notified to the process engine by means of *activity events*. In addition to these internal activity events, the process engine perceives or is notified of external *events* that are raised by other participants of the process. Case data is a collection of *data elements* that are provided or retrieved either internally through queries on the application and database layer or from other participants. In the latter case, the process engine keeps track of the participant that has provided a particular data element.

B. A process description metamodel

The main building block of a *process description* is a *decision point*. The event types, activity event types, and activity types that are relevant at a certain decision point are associated with it. In addition, a decision point consists of a set of *process rules* that capture the precise semantics of the state transitions. Process rules consist of a set of conditions which involve multiple events and data elements, and a set of conclusions which involve activities to be initiated. Via *data element descriptions* the ontology allows to model the data elements that are required at each decision point, such that missing data elements can be retrieved dynamically from specific participants. Likewise *notification descriptions* describe which participants should be notified of activity state transitions.

C. The case for process rules

Through the years a plethora of process languages and modeling constructs have been introduced. The notion of business events, for instance, goes back to the ongoing research in the area of Event-Driven Process Chains [12]. Traces of this work are present in metamodels such as the OMG's Enterprise Collaboration Architecture process metamodel [13]. Likewise many modeling constructs have a grounding in logic programming, such as Event-Condition-Action (ECA) rules, which have been shown before to represent business processes [14]. Only recently, the semantic web community has constructed a process model to fit into the current semantic web ontology language [15]. The rule-base process model, nonetheless, has some distinct features that enhance expressivity and comprehensibility over existing process models.

Expressivity is enhanced, because process rules are able to express composite events [6] and are able to deal with long-running activities. Like ECA rules, process rules can be transformed into a set of definite Horn clauses that can be used to simulate reaction rules in an inference engine, as we will show in section VI. In addition, our process model has a strong focus on case data, allowing to keep track of the origin of each data element and to determine case data requirements at each decision point. The advantage of this approach is that the content of messages no longer needs to be determined a-priori. Instead, a dynamic dialogue between participants becomes possible, consisting of event notifications, activity requests and data queries. Moreover, the focus on data elements and the rule-based context facilitates the integration of the process model with rule-based vocabulary semantics.

Enhancing expressivity by adding fine-grained modeling constructs often goes at the expense of comprehensibility. For instance, we recognize that even a small number of process rules can be difficult to verify and validate. To overcome this problem, the process model modularizes process rules into decision points. In the following section, we will describe a rule set ontology that can be used for constructing and representing rule sets concisely.

V. A RULE SET METAMODEL

A. Rule sets as the atomic units of logic

Rules are sometimes regarded as the atomic units of logic in enterprise models. In many cases, however, rule sets rather than rules should be seen as the atomic units of logic. First, to enable verification and validation, rules pertaining to a single decision regarding the vocabulary model or process model need to be modularized in rule sets. For instance, the grouping of rules that capture the discount policy of a retailer into one rule set, provides an overview of all the discount cases. Likewise, grouping process rules that pertain to a single process decision point, enables to verify and validate the semantics. A second reason why rules are not atomic is that groups with different numbers of rules can be shown to have equivalent semantics; moreover it is often the case that individual rules are dependent of one another, for instance in the context of default logics.

For these reasons it is often useful to group rules like derivation rules, conditional constraints or process rules that define the semantics of the same data element type or that pertain to the same process decision point in one rule set. In such a rule set, rules can be formulated using, for instance, a form of default logic, and transformed into equivalent sets of rules that can be more easily verified and validated, or that can be more easily executed by inference engines.

B. Decision tables

Through the years many visualizations of rule sets have come into existence, refer to [16] for an overview. For the purpose of rule-based business process modeling, we use decision tables as a graphical formalism mainly to visualize derivation and process rule sets. Consequently, we base our rule set metamodel on decision tables. We recognize, however, that not all kinds of business rule can be conveniently captured using the decision table paradigm. Figure 7 displays an example of a decision table, which is commented on in section VI. Graphically, a decision table consists of four quadrants. The two upper quadrants make up the condition sphere and the two lower quadrants represent the conclusion sphere. Likewise, the left two quadrants of a decision table make up the abstract sphere with condition labels and conclusion values whereas the right two quadrants make up the concrete sphere with conditions and references to conclusions. Properly built decision tables contain columns in the upper right quadrant that consist of a conjunction of conditions and are exhaustive and mutually exclusive. It is useful to consider decision tables as a transformation of input rules into table rules. A set of input rules, possibly expressed using a kind of default logic, determine which combination of conditions in the upper right quadrant leads to which conclusion values in the

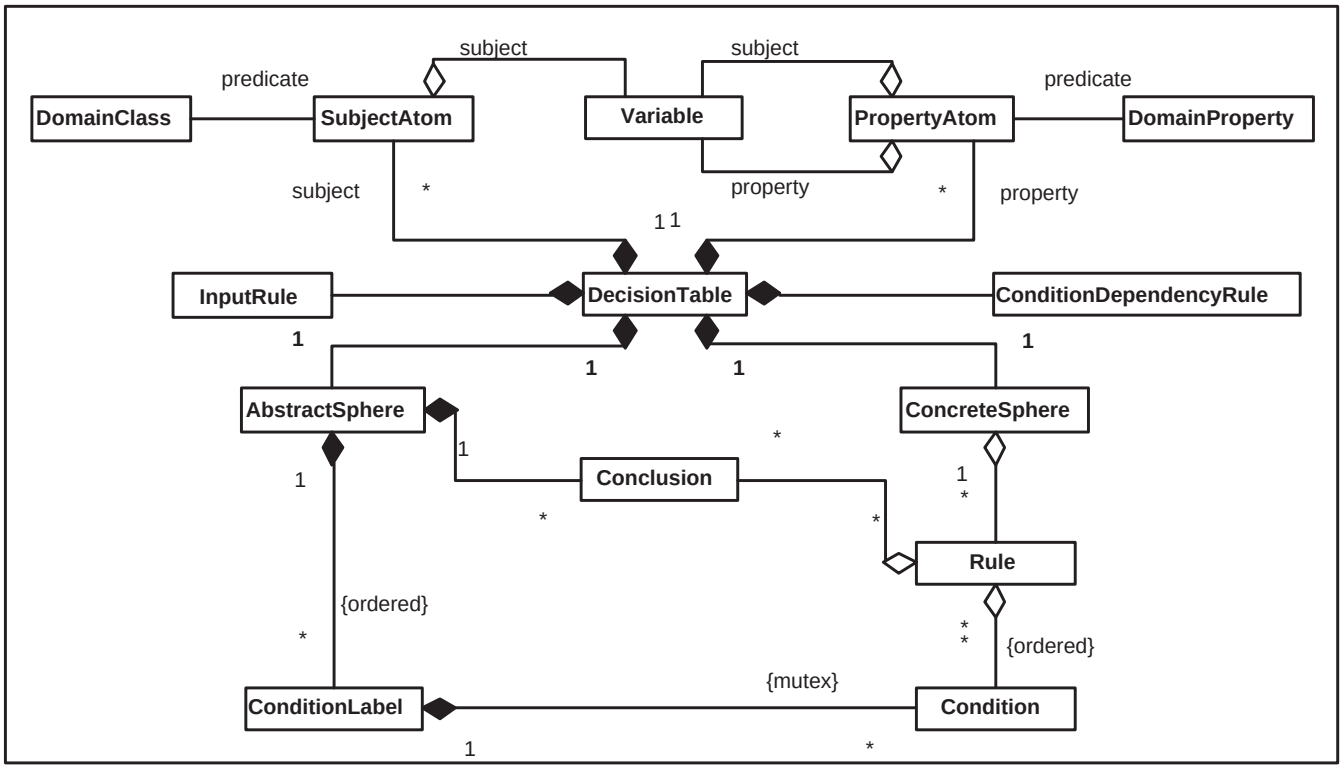


Fig. 4
A DECISION TABLE RULE SET METAMODEL

lower right quadrant.

C. A decision tables rule set metamodel

Figure 4 displays a MOF/UML decision table rule set metamodel that captures these ideas. A decision table rule set consists of subjects, properties and variables, which are represented by *subject atoms* and *property atoms*. Furthermore a rule set consists of an ordered composition of *condition labels*. These condition labels group the conditions of the rule set in sets of exhaustive and mutually exclusive conditions. Each *condition* is a logical formula that refers to the domain atoms and variables of the rule set. The actual *rules* of the rule set are an ordered conjunction of conditions, such that a rule contains at most one condition of each condition label. Notice that not every conjunction of conditions is necessarily meaningful. In other words, it might be the case that a specific condition is only meaningful in combination with other specific conditions. To express this dependency, a decision table can make use of so called *condition dependency rules*. Figure 6 displays some examples of such condition dependency rules, which are commented on in section VI. In addition to conditions, a rule refers to one or more *conclusions*. These table *rules* can be considered as a transformation of an equivalent set of *input rules*.

This generic rule set metamodel can be specialized to model business rules such as conditional constraints, derivations or process rules. In the next section we show how to generate sets of exhaustive and exclusive Horn clauses from decision tables.

VI. GENERATING RULES

As a proof-of-concept we display excerpts from a vocabulary, process and rule model using concise visualizations of the above described process and rule set metamodels and show how to generate Prolog clauses from it that can be used in the execution of the modeled process. Note that it is possible to generate rules in any logic programming language. Decision table rule sets, in particular, aim at transforming rules into exhaustive and mutually exclusive, definite Horn clauses, which have equivalent procedural semantics in many different rule execution environments.

As shown, rule-based process execution decomposes the processing of event messages into a number of logic queries on an inference engine. Each of these queries refers to a different type of rule, which are to be derived from the business vocabulary, business process and business rule models. In Prolog, these queries have the following signature:

0. **Derivation rules:** derivation rules represent properties that can be derived deductively from the available case data. Given the limited vocabulary language of domain classes and domain properties that we departed from, the signature of derivation rules in Prolog can be represented as unary and binary predicates: :ClassPredicate(?Resource), :PropertyPredicate(?Resource, ?Resource) and :PropertyPredicate(?Resource, ?Literal).
1. **Authorization rules:** authorized(+Event, +Participant) succeeds when the participant is authorized to raise the event. Likewise, authorized(+DataElement, +Participant)

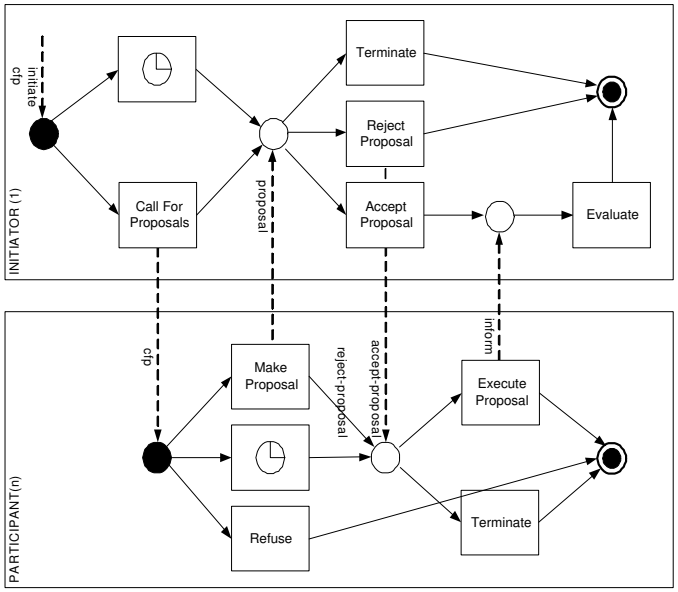


Fig. 5
THE CONTRACT NET INTERACTION PROTOCOL

succeeds if the data element description of the data element, allows it to be provided by this participant.

2. **Input validation rules:** constraintViolation(-DataElement, -ViolatedConstraint) succeeds if, given the current and additional case data elements, one of the new data elements violates a particular business constraint.
3. **Case data requirements:** requirement(+DecisionPoint, -DataElementType, -Participant) succeeds if additional data elements are required from a participant to continue the process beyond a certain decision point. If this is not the case, these data elements should be acquired from the proper process participants or underlying database and application layer.
4. **Process rules and activity preconditions:** reaction(+Event, -Activity) succeeds if given the new event and the events, data elements and activity states that make up the state of the process instance, a new activity, satisfying the preconditions of the service to invok, should be launched. The activity type and parameters of this activity are asserted in the knowledge base.
5. **Notification rules:** notification(+Activity, -Notification) succeeds if the state change of an activity requires some process participants to be notified. The proper participant, event and data elements to be contained by this notification message are asserted in the knowledge base.

By means of illustration, we derive a private process description from the Contract Net Interaction Protocol [17]. This interaction protocol is not only a standard multi-agent coordination mechanism [18], but also is a standard interaction pattern for many real-life B2B transactions. The interaction proceeds as follows. An initiator puts out a contract to tender by issuing a request for proposal (RFP). Subscribed participants can respond to this request with a proposal. In our example the initiator accepts the best proposal if it is below an a-priori determined reservation price and rejects the other proposals. This decision is taken the moment a deadline has passed. Proposals submitted after the deadline are not considered.

A. Visualization of a process description

Figure 5 displays this process description both for the initiator and for a participant. Although some visual elements correspond to other process notations, our notation is different, mainly because it distinguishes decision points in a process description, which are represented as circles in the diagram. The diagram displays activity types as rectangles. External event types are dotted lines, whereas activity event types are solid lines flowing into a decision point. The outflow of a decision point describes the activities that might be launched in response to observed events. Notice that within a single process instance multiple instances of reject proposal activity types can be created. Synchronization of these activities, however, is not required.

B. A visualization of a process rule set

The state transitions at a certain decision point are described by a set of process rules. Consider, for example, the second decision point in the process description of the initiator. At this point, a decision has to be made which proposals to accept and which to reject. In the case no proposals are submitted before the deadline, or in the case previous activities did not succeed, the process instance should terminate.

These process semantics are defined by a set of process rules, which can be visualized with decision tables. Because these tables are cumbersome to construct manually, we have used Prologa [19] to construct it for us. Although Prologa fits in a propositional logic framework, Prologa proves a useful tool to visualize and transform a number of input rules, expressed using Prologa's own default logic [19]. In particular, Prologa provides the user with a number of features that facilitate knowledge modeling, such as the reordering of condition labels to expand or contract the table, the syntactical verification of rules and the powerful visualizations that allow to semantically validate a rule set [20].

Figure 6 gives an outline of the subjects, properties and variables that are the building blocks of the process rules. The conclusions of the rule set are defined by a set of input rules. In addition a set of condition dependency rules indicates which combinations of conditions are meaningless in this table. For instance, the combination of a call for proposal succeeded event and a call for proposal failed event does not occur in reality. In consequence, these combinations of conditions are eliminated from the decision table. The actual process rules can be retrieved from the columns of the two right quadrants of the table. These rules translate in the following Prolog clauses.

```
% DECISION POINT: decisionPoint2
% PROCESS RULE: processrule2-1
reaction(Event, Activity) :-
 % EVENT
( Event=EDL; Event=ECFPs),
 % EVENT CONDITIONS
etTimerActivitySucceeded(EDL),
etCFPsucceeded(ECFPs),
 % CONDITIONS
proposal(P),
```

reaction(Event, Activity)							
1. etTimerActivitySucceeded	Y						N
2. etCFPsucceeded	Y				N		-
3. etCFPfailed	-				Y	N	-
4. proposal	Y			N	-	-	-
5. bestProposal	Y		N	-	-	-	-
6. price, reservationPrice	Pr <= RP	Pr > RP	-	-	-	-	-
1. atAcceptProposal, P	x	.	.	-	-	-	-
2. atRejectProposal, P	.	x	x	-	-	-	-
3. atTerminate	.	.	.	x	x	-	-
	1	2	3	4	5	6	7

Subjects , properties and variables :

- etTimerActivitySucceeded: EDL
- etCFPsucceeded: ECFPs
- etCFPfailed: ECFPf
- proposal: P
  - price: Pr
- bestProposal: P
- callForProposals: CFP
  - reservationPrice: RP

Condition dependency rules:

1. Condition etCFPsucceeded(ECFP) is possible only if not etCFPfailed(ECFP).
2. Condition etCFPfailed(ECFP) is possible only if not etCFPsucceeded(ECFP).
3. Condition bestProposal(P) or not bestProposal(P) is possible on ly if proposal(P).

Input rules:

4. atAcceptProposal or atRejectProposal or atTerminate is possible only if etTimerActivitySucceeded (EDL) and (etCFPsucceeded(ECFP) or etCFPfailed(ECFP)).
5. only atTerminate definitely if etDeadLine(EDL) and etCFPfailed(ECFP).
6. only atTerminate definitely if etDeadLine(EDL) and etCFPsucceeded(ECFP) and not proposal(P).
7. atRejectProposal generally if proposal(P) and not bestProposal(P).
8. atAcceptProposal generally if proposal(P) and bestProposal(P) and Pr <= RP.
9. atRejectProposal gene rally if proposal(P) and bestProposal(P) and Pr > RP.

Fig. 6
A PROCESS RULE DECISION TABLE

```
bestProposal(P),
price(P, Pr),
callForProposals(CFP),
reservationPrice(CFP, RP),
Pr=<RP,
 % ACTIVITY ASSERTION
[...]
% PARAMETER ASSERTION
[...]
 % ACTIVITIES
( Activity=AacceptProposalGUID).

% DECISION POINT: decisionPoint2
% PROCESS RULE: processrule2-4
reaction(Event, Activity) :-
 % EVENT
( Event=EDL; Event=ECFPs),
 % EVENT CONDITIONS
atTimerActivitySucceeded(EDL),
etCFPsucceeded(ECFPs),
 % CONDITIONS
not(Proposal(P)),
 % ACTIVITY ASSERTION
[...]
% PARAMETER ASSERTION
[...]
 % ACTIVITIES
( Activity=AterminateGUID).
```

discount(Contract, D) – MAX D					
1. inDelayWithPayment	Delay=true	Delay=false			
2. quantity	-	Q >= 3		Q < 3	
3. deliveryTime	-	T < 7	T >= 7	T < 7	T >= 7
1. D is 0.1*SPrice	-	x	x	.	.
2. D is 50	-	.	x	.	x
3. D is 0	x	x	x	x	x
	1	2	3	4	5

Subjects , properties and variables :

- salesContract: Contract
  - standardPrice: SPrice
  - quantity: Q
  - deliveryTime: T
  - discount: D
  - customer: Customer
- customer: Customer
  - inDelayWithPayment: Delay

Input rules:

1. D=0 generally always.
2. D =0.1* SPrice if Q >= 3.
3. D=50 if T >= 7.
4. Only D=0 definitely i f Delay=true.

Fig. 7
A DEDUCTION RULE DECISION TABLE

C. A visualization of a derivation rule set

Decision tables are useful visualizations of conditional constraints, derivation rules and process rules. In the next example we show that this table is also capable of expressing higher-order concerns such as aggregation. Figure 7 displays the discounts a retailer might attribute to a sales contract, for instance, in response to a request for proposal. From the table it is clear that in some cases, a sales contract might qualify for multiple discounts – this is called a multiple-hit table in the literature. Suppose the retailer has the policy to always grant the highest discount to a sales contract. The retrieval of the highest conclusion value, might translate in the following Prolog clauses.

```
discount(Contract, Discount) :-
 salesContract(Contract),
 findall(D, discount_sub(Contract, D), Discounts),
 max(Discounts, Discount).

% DERIVATION RULE: discount1
discount_sub(Contract, D) :-
 salesContract(Contract),
 customer(Contract, Customer),
 inDelayWithPayment(Customer, true),
 D is 0.
[...]
% DERIVATION RULE: discount5
discount_sub(Contract, D) :-
 salesContract(Contract),
 customer(Contract, Customer),
 inDelayWithPayment(Customer, false),
 quantity(Contract, Q),
 Q < 3,
 deliveryTime(Contract, T),
 T >= 7,
 (D is 0;
 D is 50).
```

VII. CONCLUSION

In this paper, we set out for using a business rule model to represent the semantics of the data and control-flow perspective of business processes. In particular, we have shown how a business rule model defines and constrains the data elements of a business vocabulary model and the state transitions of a business process model. To this end we have constructed a generic,

lightweight process and rule set metamodel that can be the basis of process languages of arbitrary complexity. From these models a corpus of different kinds of definite Horn clauses are to be generated, which can be used in the execution of the modeled process. In addition we have shown how rule-based process execution decomposes the processing of event messages in a number of logic queries on an inference engine. The latter indicates how to embed business logic in a service-oriented, event-driven architectural framework that links the enforcement of rules and the orchestration of services.

At the moment we are continuing our research in the following three directions. First of all, we are working on a rule generator to automatically generate the execution-level rules from Semantic Web-based vocabulary, process and business rule models. Secondly, we are looking into process validation facilities for the validation of private process descriptions against predefined public interaction protocols. In addition we envision to extend the process metamodel to allow for the incorporation of update and delete case data manipulation facilities both for the process engine and the process participants.

REFERENCES

[1] Jens Dietrich, Alexander Kozlenkov, Michael Schroeder, and Gerd Wagner, "Rule-based agents for the semantic web.," *Electronic Commerce Research and Applications*, vol. 2, no. 4, pp. 323–338, 2003.

[2] Maja D'Hondt and Viviane Jonckers, "Hybrid aspects for weaving object-oriented functionality and rule-based knowledge," in *AOSD '04: Proceedings of the 3rd international conference on Aspect-oriented software development*, New York, NY, USA, 2004, pp. 132–140, ACM Press.

[3] Rachid Hamadi and Boualem Benatallah, "A Petri net-based model for web service composition," in *CRPITS'17: Proceedings of the Fourteenth Australasian database conference on Database technologies 2003*, Darlinghurst, Australia, Australia, 2003, pp. 191–200, Australian Computer Society, Inc.

[4] L. G. Meredith and Steve Bjorg, "Contracts and types," *Commun. ACM*, vol. 46, no. 10, pp. 41–47, 2003.

[5] Kuldar Taveter and Gerd Wagner, "Agent-Oriented Enterprise Modeling Based on Business Rules.," in *ER*, Hideko S. Kunii, Sushil Jajodia, and Arne Sølvberg, Eds. 2001, vol. 2224 of *Lecture Notes in Computer Science*, pp. 527–540, Springer.

[6] James Bailey, Franois Bry, and Paula-Lavinia Pătrânjan, "Composite event queries for reactivity on the web," in *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, New York, NY, USA, 2005, pp. 1082–1083, ACM Press.

[7] Lawrence Wilkes and Richard Veryard, "Service-Oriented Architecture: Considerations for Agile Systems," *Microsoft Architects Journal*, pp. 11–23, April 2004.

[8] Marko Bajec and Marjan Krisper, "A methodology and tool support for managing business rules in organisations," *Information Systems*, vol. 30, no. 6, pp. 423–443, Sep 2005.

[9] Terry A. Halpin, "A Fact-Oriented Approach to Business Rules," in *ER*, 2000, pp. 582–583.

[10] Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker, "Description logic programs: combining logic programs with description logic," in *WWW '03: Proceedings of the 12th international conference on World Wide Web*, New York, NY, USA, 2003, pp. 48–57, ACM Press.

[11] W. M. P. Van Der Aalst, A. H. M. Ter Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow Patterns," *Distrib. Parallel Databases*, vol. 14, no. 1, pp. 5–51, 2003.

[12] G. Keller, M. Nüttgens, and A.W Scheer, "Semantische Prozessmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK)," Technical Report 89, Institut für Wirtschaftsinformatik Saarbrücken, Germany, 1992.

[13] Object Management Group, "UML Profile for Enterprise Collaboration Architecture Specification," Februari 2004, Document number 04-02-05.

[14] Holger Herbst, "A Meta-Model for Business Rules in Systems Analysis," in *CAiSE*, Juhani Iivari, Kalle Lyytinen, and Matti Rossi, Eds. 1995, vol. 932 of *Lecture Notes in Computer Science*, pp. 186–199, Springer.

[15] "OWL-S: Semantic Markup for Web Services," 2003.

[16] Grigoris Antoniou, Kuldar Taveter, Mikael Berndtsson, Gerd Wagner, and Silvie Spreeuwenberg, "A First-Version Visual Rule Language," Report IST-2004-506779, REWERSE, 8 2004.

[17] Reid G. Smith, "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver.," *IEEE Trans. Computers*, vol. 29, no. 12, pp. 1104–1113, 1980.

[18] FIPA, *FIPA Contract Net Interaction Protocol Specification*, FIPA, 2002.

[19] Jan Vanthienen, *Prologa 5.2 - tabular knowledge modeling - User's manual*, Katholieke Universiteit Leuven, 2003.

[20] Jan Vanthienen, Christophe Mues, and Ann Aerts, "An Illustration of Verification and Validation in the Modelling Phase of KBS Development.," *Data Knowl. Eng.*, vol. 27, no. 3, pp. 337–352, 1998.

Author Index