

Babes-Bolyai University
Faculty of Economics and Business Administration
Business Information Systems Department

**New Methodology and Complete Set of Tools for
DSL Conceptualization
-case study on developing SCOR-based modeling tool-
~Thesis Summary~**

Scientific Advisor:

PhD. Ștefan Ioan Nițchi, Professor

PhD. Student:

Costin Aurelian Răzvan

-2013-

Thesis Table of Contents

Introduction.....	
1 Chapter 1: Formal Language Definition.....	
1.1 Introduction	
1.2 Definitions.....	
1.3 Language textual definition.....	
1) The set of attributes and the notation composition rule	
2) The set of notations	
3) The set of relations	
4) The set of contexts.....	
5) The set of adverbs	
1.4 Formal definition of a language	
1.5 Graph based conceptualization of a language	
1.6 Related work	
1.7 Concluding Remarks for Chapter 1.....	
2 Chapter 2: ML ² Language for DSL Modeling.....	
2.1 Introduction	
2.2 ML ² Detailed Syntax.....	
2.3 ML ² vs. MOF M3	
2.3.1 Comparative analysis of UML modeling language models.....	
2.3.2 Other approaches for the identified MOF M3 shortcomings.....	
2.4 Concluding Remarks for Chapter 2.....	
3 Chapter 3: Language Engineering Tool.....	
3.1 Introduction	
3.2 Language Engineering Tool Description	
3.3 Reengineering Module	
3.3.1 Case Study on VLML and Discussions	
3.4 Language Description Module.....	
3.5 Language Analysis Module.....	
3.6 Concluding Remarks for Chapter 3.....	
4 Chapter 4: Supply Chain Operations Reference.....	

4.1	Introduction
4.2	SCOR Processes
4.3	SCOR Metrics
4.4	SCOR in Practice
4.4.1	HP
4.4.2	BMW Motorrad
4.4.3	STATOIL
4.5	SCOR vs. VRM.....
4.5.1	Feature based comparison.....
4.5.2	Comparison of Framework Architectures.....
4.5.3	Process Analysis Based Comparison
4.5.4	Application based comparison.....
4.5.5	Concluding remarks related to section 4.5.....
4.6	Concluding Remarks for Chapter 4.....
5	Chapter 5: Development of a new SCOR Modeling Tool.....
5.1	Introduction
5.2	SCOR Modeling Software identification
5.3	Criteria catalog
5.4	Requirements for SCOR modeling tools.....
5.5	Software analysis.....
5.5.1	Aris EasyScor
5.5.2	Tibco Business Studio
5.5.3	Cogniti
5.5.4	Metastorm ProVision
5.5.5	Nimbus Control
5.5.6	Xelocity Process Wizard
5.5.7	Software Analysis Overview
5.6	SCOR-based Modeling Language Conceptualization.....
5.6.1	Product model
5.6.2	Scope model.....
5.6.3	Sales region.....

5.6.4	Supply regions
5.6.5	Geographical map
5.6.6	Extended thread diagram
5.6.7	Business Process model
5.6.8	Organizational model.....
5.6.9	SCOR-based DSL Overview
5.6.10	Supply Chain Modeling Workflow.....
5.7	Concluding Remarks for Chapter 5.....
Summary and Future Work.....	
1	Contributions.....
2	Future work.....
Bibliography	

Abstract

The contribution of the thesis is threefold. Firstly, starting from an innovative definition of a language as a concept, based on a simple hypothesis that states that a word is defined by a unique set of attributes, the authors construct a theory explaining every element of a language, the relations between these elements, and how statements are constructed. The formal definition of a language could answer to a lot of natural language processing (NLP) problems. Secondly, based on these definitions, authors describe the methodology of conceptualizing languages, and propose a tool meant to help language conceptualizers to describe new domain specific languages (DSLs). The tool allows its users to engineer, reengineer, analyze, and compare DSLs. The goal of this tool is to provide insight about languages. The outcome of the engineering module is a data set which can be easily transformed and imported in meta-modeling platforms to generate new modeling tools. The graph that is generated following the conceptualizing process describes the language in detail and it can be used to pass requirements to people in charge of implementing new DSLs or to validate the DSLs that were implemented on meta-modeling frameworks. Lastly, the thesis describes a case study on conceptualizing a new modeling tool for supply chain process management, analysis, simulation, and optimization. The modeling tool will be constructed based on supply chain operations reference (SCOR) and the conceptualization phase will be conducted using the proposed Language Engineering tool.

Introduction

In this thesis, we propose innovative approach in three different research fields: economy, informatics, and linguistics. The described findings are the result of a research period that implied scaling up and down the abstraction levels which obliged us to study parts from all these domains.

The economical field was touched due to the main goal of the research which was to develop a modeling tool for supply chain processes.

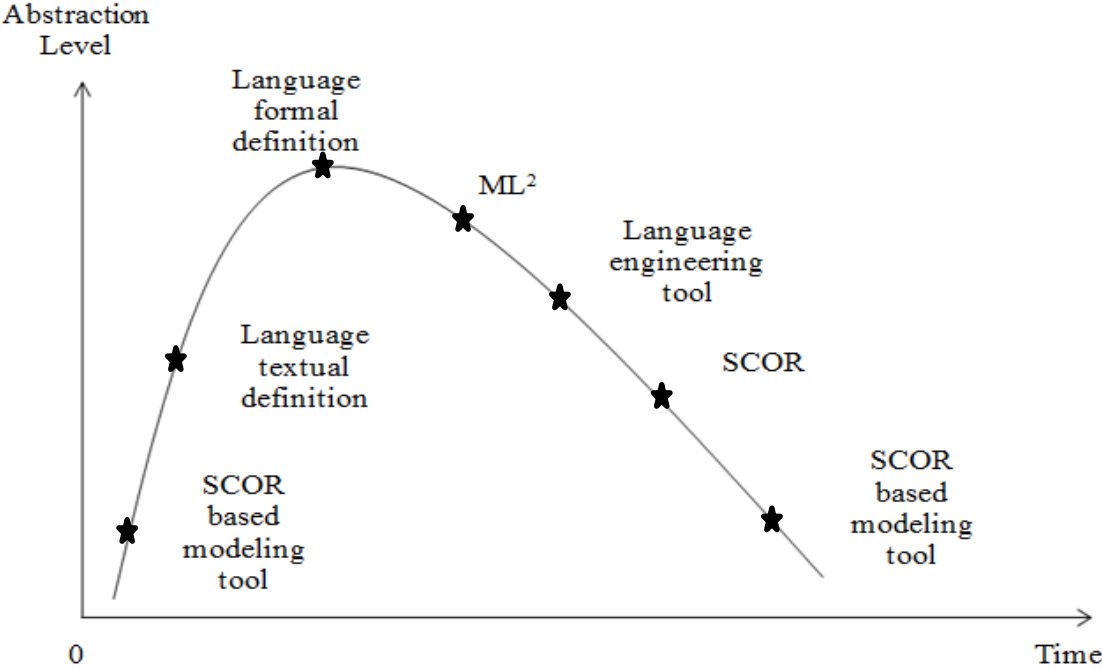
Linguistics was studied as we had to provide an innovative definition of a language as a concept, which could explain how languages are conceptualized.

Informatics was approached as we propose a tool meant to help language conceptualizers to describe new domain specific languages (DSLs). The tool allows its users to engineer, reengineer, analyze, and compare DSLs. The goal of this tool is to provide insight about

languages. The outcome of the engineering module is a data set which can be easily transformed and imported in modeling platforms to generate new modeling tools. The graph that is generated following the conceptualizing process describes the language in detail and it can be used to pass requirements to people in charge of implementing new DSLs or to validate the DSLs that were implemented on meta-language frameworks.

When the research was started, its main goal was to provide a tool that will enable its users to model, analyze, simulate, optimize, and manage supply chains. Along the way, it was discovered that, to be able to reach this goal, a deeper and more abstract research on language definitions and conceptualization is required. Thus, we scaled up on the abstraction hierarchy and tried to get more insight regarding language definitions, structure, and conceptualization methodology. After this foray in the abstraction layers, the findings were used to conceptualize a supply chain modeling tool, which, basically, was the initial research goal.

We can graphically represent this research evolution using a Gauss chart:



When the goal of the research was set, we had few ideas on how a supply chain modeling tool should look like as we lacked knowledge regarding language conceptualizations methodology. While the technical part involved in the research was a smaller issue, the conceptualization of the language which shall be used to model supply chains, the technique of supply chains modeling, and the algorithms for supply chains process simulation and analysis were a totally unknown

field. Therefore, we had to climb up the abstraction levels and try to define and understand languages as concepts. The main assumption was that, if a language structure could be graphically represented, than it should be possible to develop a tool for language modeling, which could be used, then, to conceptualize a language from scratch.

The first step we took was to textually describe a language based on recommended literature and state of the art. Unfortunately, every chosen track, led us to dead ends, as we could not use existing language definitions to understand the structure of a language. Thus, as it can be seen on the chart above, we decided to write our own theory about language. The next step was to compile a formal definition of the language as concept. This step represents the highest abstraction level reached. The purpose of this stage was to identify the mechanism that stands behind a human or machine that is able to conceptualize a language. The fundamental questions of the research, at this point were: “How does a word appear?”, “What triggers the necessity of conceptualizing a new word?”, “What is the real meaning of this word?”. The assumption was that a word is describing a set of attributes, and, by an attribute, we understood a perception of a language conceptualizer. More on this theory is detailed in Chapter 1.

After the formal description of a language, we had to demonstrate ourselves that this definition is valid. Thus, the language for language conceptualization (which was named as ML^2) appeared. ML^2 stands for Modeling Language for Language Modeling. Basically, we have conceptualized an auto-descriptive language that would allow model engineers to describe new DSLs. The language was given to model engineers to reengineer some existing modeling languages. The results are listed in Chapter 2. As these results have been very encouraging, the first module of the engineering tool was developed. The tool had the purpose to automatically reengineer a DSL implemented in a modeling platform. The expected results of this module were a graphical representation of that DSL, using ML^2 , together with the statistics regarding the number of words, possible statements, erroneous sub-structures, etc.

The results kept being encouraging, thus, the tool was enhanced. Two new modules were developed and integrated in to the Language Engineering Software: an analysis module meant to validate an implemented DSL, and an engineering module that can be used to conceptualize a language, based on a common approach and methodology. The Language Engineering Software is presented and detailed in chapter 3.

The last part of the research was dedicated to accomplish the initial goal, and that was to develop a supply chain modeling language. Having the needed knowledge, and a complete set of tools, which could be used to conceptualize the supply chain modeling language, we have based our work on well-known existing frameworks like VRM and SCOR and described the new modeling tool. The results of this part of the research are listed in Chapter 4 and 5.

1 Chapter 1: Formal Language Definition

Ferdinand de Saussure was the first author who made the distinction between “language” and “langue” which was globally accepted [3]. He states that a “langue” is, in fact, an instance of a language. From de Saussure point of view English or French or Romanian etc. are “langues” while “language” itself is the concept that describes all “langues”. Thus, the definition of a “langue” may state that “it is a language that (...)”. But, then, what is a language? In this thesis, we opine that, *the language is a set of words, conforming to a rigid structure and a set of rules that if applied on the set of words, allows statements to be created*. Basically, this definition is the subject of this section, as we shall cleave it in order to demonstrate it.

Ferdinand de Saussure’s thesis shows that a simple reasoning would imply that all “langues” are “language”. The “langues” can be further divided in well-known categories as: natural languages (ex: English, French, Romanian, etc.), Domain Specific Languages [9], programming languages (ex: C, C++, Java, etc.), mathematics (ex: algebra, logics, etc.), etc. Thus we can say that all types of “langues” are “language”.

The authors propose the following schema to distinct between “language” as a concept, “language type” as a class that contains languages that are related, like natural languages, programming languages, domain specific languages, etc., and all the specific linguistic systems.

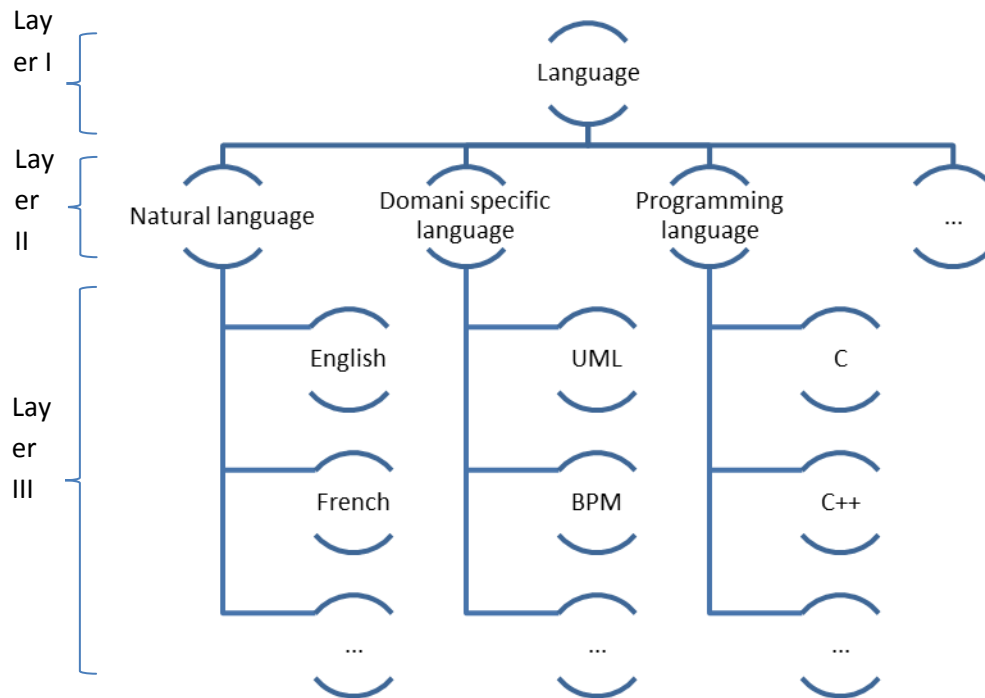


Figure 1 Language hierarchy

The 1st layer comprises only one node: “language”. At this level of abstraction the language should be perceived as a concept that defines all types of languages. The language at the 1st layer is the subject of this chapter.

We have defined language in the first paragraph of this section. We may extend this definition and say that at the 1st level of abstraction a language is defined by a set of language elements and a set of rules that governs the statement creation. This definition shall be true for all the 3rd level languages. If we divide the set of language elements in 5 sub-sets: a set of attributes, a set of notations, a set of relations, a set of contexts, and a set of adverbs; we may say that the set of rules that governs the statement creation process in any 3rd layer language comprises 5 basic rules: notation composition rule, relation describing rule, subject binding rule, object binding rule, and context allocation rule, and conclude that this is the definition of a language structure.

1.1 Formal definition of a language

Definition: A language is a 10-tuple= $\langle A1, N, R, C, A2, \alpha, \omega, s, o, \varepsilon \rangle$ where:

- A1 is a finite set of attributes;
 - The authors emplace “U:U” to represent the “empty attribute”. The empty attribute is an attribute that has no data type or value.
 - As a consequence the following relation $A1 \cup \{U:U\} = A1$ is defined.

- The attribute that has a data type and no value will be represented as “Any name”: U.
- $A1 \cup \{ \text{“Any name”}: U \} = A$ where A contains all the elements of A1 and the (“Any name”: U) element.
- N is a finite set of notations;
 - The authors emplace “U:U:{U:U}” to represent the “empty notation”. The empty notation is a notation that has no name, no representation, and it is associated to a set of attributes that contains only the empty attribute.
 - As a consequence the following relation $N \cup \{ U: U: \text{Attr}\{U:U\} \} = N$ is defined.
 - The authors call the empty notation that has a name the “incipient element”. Thus “Any Name”: U: Attr{U:U} is the incipient element of the language.
 - $(\text{“Any Name”}: U: \text{Attr}\{U:U\}) \in N$.
- R is a finite set of relations;
 - R does not have an empty element.
- C is a finite set of contexts;
 - C does not have an empty element.
- A2 is a finite set of adverbs;
 - The authors emplace “U:U” to represent the “empty adverb”. The empty adverb is an adverb that has no data type or value.
 - As a consequence the following relation $A2 \cup \{ U:U \} = A2$ is defined.
 - The adverb that has a data type and no value will be represented as “Any name”: U.
 - $A2 \cup \{ \text{“Any name”}: U \} = A$ where A contains all the elements of A2 and the (“Any name”: U) adverb.
- “ α ” is a function associated with the “notation composition rule”. The function is defined as, $\alpha: A1^m \rightarrow N$ where $m \leq c$ (the A1 cardinality) is an integer. The function associates a set of attributes to a notation.

The function “ α ” has the following properties:

1. α is surjective, meaning: every notation has a corresponding combination of attributes in A1.
2. If two or more notations are associated to the same combination of attributes one may say that the notations are synonyms.

If one considers α bijective, then the language will not allow synonyms and every possible combination of attributes will describe a notation. One may call this language a “complete language”. If α is bijective then one can define $\bar{\alpha}$ as the inverse of α . Thus, the authors set the following property to $\bar{\alpha}$:

1. $\bar{\alpha}=\alpha^{-1}$; $\bar{\alpha}: N \rightarrow A1^m$ associates a set of attributes to a notation.

- “ ω ” is a function associated to “relation describing rule”. The function is defined as, $\omega: A2^m \rightarrow R$ where $m \leq c$ (the A2 cardinality) is an integer. The function associates a set of adverbs to a particular relation.

ω properties:

1. ω is surjective, meaning: the same relation can be described by different sets of adverbs. For example, the relation “runs” might be described by {speed: fast} or by {distance: 11km}.
2. A relation that has no adverbs will be associated to {U:U} the empty adverb set.

- s is a function associated to “subject binding rule. The function is defined as $s: N^*R \rightarrow S$. The function associates a pair composed by a notations and a relation to a set S . The pairs in the set S are meant to show which notations can be subjects of which relations.

s properties:

1. s is surjective, meaning every element in S has an element of N^*R associated. But not every element in N^*R can be eligible to be in S . S contains pairs that are specified by the language conceptualizer with the purpose of showing that a notation may be the subject of a relation.
2. One cannot have a pair as $((U: U: Attr\{U: U\}), Rx)$ where $U: U: Attr\{U: U\}$ is the blank notation and Rx is a relation in the R set, as one cannot define relations that have no possible subjects.

- o is a function associated to “object binding rule”. The function is defined as $o: N^*R \rightarrow O$. The function associates a pair composed by a notation and a relation to a set O . The pairs in the set O are meant to show which notations can be objects of which relations.

o properties:

1. o is surjective, meaning every element in O has an element of N^*R associated. But not every element in N^*R can be eligible to be in O . O contains pairs that

are specified by the language conceptualizer with the purpose of showing that a notation may be the object of a relation.

2. One may have a pair as $((U: U: \text{Attr}\{U: U\}), R_x)$ where $(U: U: \text{Attr}\{U: U\})$ is the blank notation and R_x is a relation in the R set, as one could define relations that have no possible objects.
- ε is a function associated to “context allocation rule”. The function is defined as $\varepsilon: \text{NUR} \rightarrow C$. The function associates notations and relations to one or more contexts in the context set C where the statements built using the notations and relations have specific understandings.

ε properties:

1. ε is surjective, meaning that every context in the context set has at least two language elements associated: a notation and a relation. The purpose of the context is to imply the meaning of the statements created with the associated language elements. The meaning of the language elements changes depending on the context in which they are used. A notation or a relation can be allocated to more than one context.
2. A context must have at least one notation and one relation.

Everything that operates according to these specifications is a language. The total amount of languages belongs to the Language class.

2 Chapter 2: ML² Language for DSL Modeling

This chapter presents a new modeling language for language modeling, which we have named ML². The idea of a language for language conceptualization and modeling is not new. Relevant and well-known such languages already exist (like MOF), but domain specific language literature states that a DSL should be used by engineers who, looking at their workbench, realize they need a better drill and find that a particular DSL provides exactly that.

Model engineers face real issues when starting a new modeling method development project as they need a more specific language than the 3rd layer of MOF to conceptualize the language they will use. Concerning this need, we propose a language that allows model engineers to model a language easily.

We defined our language for modeling language developing ideas from the MOF's 3rd layer and based on a scrutiny of language reengineering in reverse engineering strategies. Regarding these

aspects, we tried to model some existing languages using the MOF approach, as being the standard for meta-meta-modeling language. Even though we were able to present the structure of a language we weren't able to specify concrete syntax and grammatical aspects. In this sense we considered that a new approach over language modeling language should be used.

To conceptualize ML² we embraced a top-down approach, starting with analyzing the domain of language modeling which allowed us to identify the following aspects that our DSL should address:

1. concrete syntax for the language elements that have no representation (classes that provide the initial framework). Example: the incipient element, a notation that has no attributes or representation (IE: Attr{U: U});
2. concrete syntax for language elements that have representation, meaning user defined classes that represent real world objects. Example: any of the notations like C: C: Attr{c:c1};
3. concrete syntax for relation (the user defined classes that represent possible actions and interactions of the real world objects).
4. concrete syntax for the contexts, namely the composite structures in which the combination of notations and relations have a meaning.
5. grammar, meaning that we defined four types of connectors between classes: inherits (which has the same meaning as in UML Class Diagram dictionary), belongs to (which is a connector that connects concrete syntax notations and relations to a specific context), subject and object connectors (which are the connectors between a notation and a relation representing the order in the statement as well).

The proposed DSL differentiates itself from UML as it is meant to tackle a wider spectrum of languages. If UML is concerned mostly on programming languages, which requires precise formalism and universal statements, our language enables a more flexible formalism, defining relations between notations as relations, that could have different meaning in case of "adverbs" association. For example, if the relation "equals" has to have a precise meaning in programming, in other languages we can say that there is a difference between an "equal" relationship and an "equal" relationship which has the "more or less" adjective associated.

In contrast with other comparable approaches, the proposed DSL features more general means to specify description of a language. Thus, our DSL should be considered more abstract and universal than UML, but in the same time specialized on a specific domain, that is modeling

languages. In the next section we will compare our language against MOF M3 which is the most used instrument for language conceptualization.

2.1 ML² Detailed Syntax

In this section we present a more detailed overview on the syntax of our proposed DSL for language modeling. Nevertheless, in the tables bellow we describe the syntactical construction of our language.

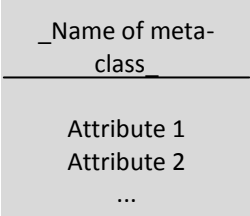
Name	"Meta Class"
Graphical Representation	
Description	The "Meta Class" is used to model the initial framework of the language. The color gray, suggests the fact that this class is a meta-class. A meta-class is a notation that has no representation.
Type	Class
Classes	It can have relations with: <ul style="list-style-type: none"> - other "Meta-Class"; - "User Defined Class"; - "Relation Class".
Connectors	Types of connectors: <ul style="list-style-type: none"> -inheritance- between 2 "Meta Class", a "Meta Class" and a "User Defined Class"; -subject, object connector- between a "Meta Class" and a "Relation Class".

Table 1 "Meta Class" class description

Name	"User Defined Class"
------	----------------------

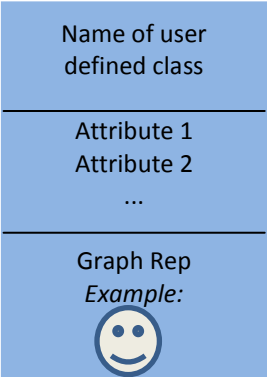
Graphical Representation	
Description	<p>The "User Defined Class" is used to model the objects from the real world. We propose, as a graphical representation of this class, a blue rectangle that is split into three sections. The first section allows the model engineer to assign a name for the object. The second section shows the attributes representing the object's "adjectives". The third section shows the graphical representation of the object in the language that is conceptualized. The color has the purpose to differentiate this class from the "Meta Class" and the "Relation Class". The "User Defined Class" is used to represent notations that have representation.</p>
Type	Class
Classes	<p>It can have relations with:</p> <ul style="list-style-type: none"> - "Meta-Class"; - "Relation Class"; - "Contexts".
Connectors	<p>Types of connectors:</p> <ul style="list-style-type: none"> -inheritance- between a "Meta Class, and a "User Defined Class"; -subject, object connector- between a "User Defined Class" and a "Relation Class"; -belongs to- between a "User Defined Class" and a "Context".

Table 2 "User Defined Class" class description

Name	"Relation Class"
------	------------------

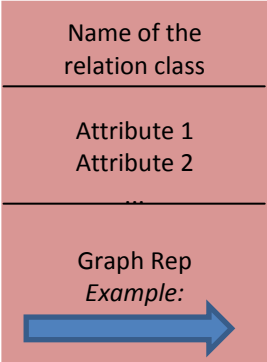
Graphical Representation	
Description	<p>The "Relation Class" is used to model actions (verbs) from the real world. As it can be seen above, we propose, as a graphical representation of this class, a red rectangle that is split into three sections. The first section allows the model engineer to assign a name for the relation. The second section shows the attributes representing the relation's "adverbs". The third section shows the graphical representation of the object in the language that is conceptualized. The color has the purpose to differentiate this class from the "Meta Class" classes and the "User Defined Class" classes. The "Relation Class" is used to represent relations.</p>
Type	Class
Classes	<p>It can have relations with:</p> <ul style="list-style-type: none"> - "Meta-Class"; - "User Defined Class"; - "Context".
Connectors	<p>Types of connectors:</p> <ul style="list-style-type: none"> -subject, object connector- between a "User Defined Class" and a "Relation Class"; -belongs to- between a "Relation Class" and a "Context".

Table 3 "Relation Class" class description

Name	"Context"
------	-----------

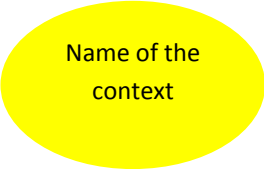
Graphical Representation	
Description	The "Context" is used to model the context in which statements have meaning. Before creating a statement the user of the conceptualized language must define the context for the statement. Thus, the model engineer that conceptualizes the language needs the means of modeling which objects and actions can be used in which contexts.
Type	Container
Classes	It can have relations with: - "Relation Class"; - "User Defined Class".
Connectors	Types of connectors: -belongs to- between a "User Defined Class" class and a "Context"; -belongs to- between a "Relation Class" class and a "Context".

Table 4 "Context" container description


Name	"inherits"
Graphical Representation	
Description	The "inherits" connector is used to build the initial structure of the language. It shows the class hierarchy and how the user defined syntax maps on an initial meta-model of the conceptualized language.
Type	Connector
Connects Classes	It can connect: - "User Defined Class" to "Meta Class"; - "User Defined Class" to other "User Defined Class"; - "Meta Class" to other "Meta Class";

Table 5 "Inheritance" connector description


Name	"belongs to"
Graphical Representation	
Description	The "belongs to" connector is used to model which "User Defined Class" or "Relation Class" can be used in which "Context". This allows the model engineer to establish contexts in which statements can be build using just the allowed elements.
Type	Connector
Connects Classes	It can connect: - "User Defined Class" to "Context"; - "Relation Class" to "Context".

Table 6 "Belongs to" connector description

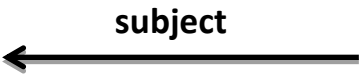

Name	"subject"
Graphical Representation	
Description	The "subject" connector is used to model which "User Defined Class" is the subject for the "Relation Class". Nevertheless, it shows the position of the subject relative to the action in a statement.
Type	Connector
Connects Classes	It can connect: - "Relation Class" to "User Defined Class".

Table 7 "Subject" connector description

Name	"object"
Graphical Representation	
Description	The "object" connector is used to model which "User Defined Class" is the object for the "Relation Class". Nevertheless, it shows the position of the object relative to the action in a statement.
Type	Connector

Connects Classes	It can connect: - "Relation Class" to "User Defined Class".
------------------	--

Table 8 "Object" connector description

3 Chapter 3: Language Engineering Tool

In this chapter we describe a Language Engineering Tool, which is software that enables users to define new languages, analyze them, validate them, and implement them in meta-modeling platforms to create new modeling tools for certain domains. Our proposed tool uses ML² for creating visual representations of languages.

The tool has three basic modules:

1. A Reengineering Module;
2. A Language Analysis Module;
3. A Language Description Module.

The Language Engineering tool was developed upon a component-based architecture. Our system deployed as a standalone java application is compounded by an input component, a meta-model structure component, a representation component, and an output module that, when receives the results from the representation component, generates different representation formats of the results.

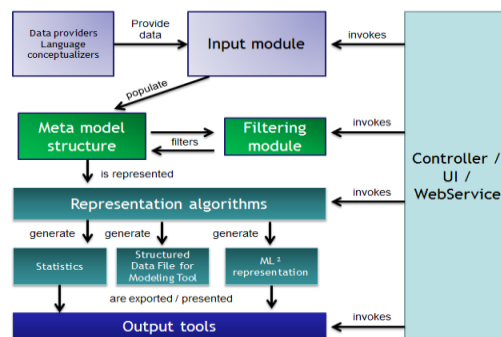


Figure 2 Language Engineering Tool Architecture – Component Overview

3.1 Reengineering Module

The Reengineering Module proposes a very simple user interface. The interface provides 5 panels. The first 4 panels (Input, Results, Image Type, and Choose Action) require user inputs, while the fifth panel is used to output a preview of the reengineering results. The use case scenario in this module can be described as follows:

1. In the first panel the user is asked to provide the type of the file which will be processed (ML2 file or ADOxx file), and then provide the file itself. The file must contain the description of the language as defined by the language conceptualizer;
2. In the second panel (Results), the user is asked to choose the type of the expected results. There are two types of results listed: (1) a “statistics” result which presents the language in a document containing a table in which the language is described (like the one used for defining the ML² language in Chapter 2); (2) and an “image overview” result which provides the ML² graph representation of the language;
3. In the third panel, the user is asked to select the type of file in which the results should be listed;
4. In the fourth panel, the user can choose to reengineer only some aspects of the language;
5. After, the user inputs all the information required, in panel five, a preview of the results is listed, and the results document is generated.

3.2 Language Description Module

We used the language engineering methodology described in Chapter 1 to enable DSL engineers to easily create a new DSL that could be then imported in a meta-modeling platform to create a new modeling tool. We based our work on ADOxx meta-modeling platform, to demonstrate the validity of the module’s functionality. Thus, the tool is meant to provide a data structure that can be imported in ADOxx to create a new modeling tool. However, the tool can be easily extended to provide data structures characteristic to any other meta-modeling platforms that provide an import DSL mechanism.

The use case of the tool is simple and straight forward. The user chooses to use the Language Description Module. In the opened user interface he starts adding elements to the language. There is no restrictions regarding the steps in which he adds the elements. It is up to the user if he starts by defining the notations, relation, attributes, or contexts.

When the user introduces all language elements in the language description form he has to save the language. At this moment no validation of the language is made, due to the fact that, meta-modeling platforms accept all kind of DSLs, not considering the DSL structure, as long as the data set in which the DSL is described conforms the technical structure required by the meta-model platform’s import engine.

If the user wants to analyze, or validate the newly created DSL's structure he can use the Language Analysis Module.

3.5. Language Analysis Module

The Language Analysis Module was developed to solve two important language engineering issues: DSL analysis and DSL comparison.

Firstly, the module uses the language formalism defined in Chapter 1 to analyze the structure of a DSL. The result of this functionality is a document containing the statistics concerning the DSL. The document contains information regarding the number of notations, relation, contexts, attributes, and adverbs. Nevertheless, the document presents which attributes are associated with which notations, which adverbs are associated with which relations, which are the valid type of statements, and discourses, and how should these be understood depending on contexts. The document also informs users if invalid language elements were declared, meaning elements that do not adhere to the imposed formalism.

Secondly, the module provides the functionality for comparing two DSLs. The idea here is to provide the user a way to validate its DSL against an initial requirement. We consider the DSL described using ML^2 as the initial requirement, while the DSL that is intended to be validated is the one provided by a meta-modeling platform, in which that DSL was deployed. The two DSLs are compared as graphs.

4 Chapter 4: Supply Chain Operations Reference

In this chapter we have analyzed supply chains with the intention of conceptualizing a new modeling tool for supply chain operations. We have chosen to base our research on a very well-known DSL in the supply chains theory, which is SCOR.

Supply chain operations reference (SCOR) is a framework for modeling and analysis that provides standard guidelines for companies to manage their supply chains. SCOR is the product of Supply Chain Council (SCC), an independent, not-for-profit, global corporation with membership opened for all companies interested in applying and sharing standardized supply chain management practices. In addition SCOR helps companies to adopt best practices where deemed appropriate [38] and therefore can be considered a normative modeling approach [39].

The SCOR model has been developed to define all business activities associated with the supply chain [40]. It spans: all customer interactions (order entry through paid invoice), all physical material transactions (supplier's supplier to customer's customer, including equipment, supplies,

spare parts, bulk product, software, etc.), and all market interactions (from the understanding of aggregate demand to the fulfillment of each order). It does not attempt to describe every business process or activity. Specifically, the Model does not address: sales and marketing (demand generation), product development, research and development, and some elements of post-delivery customer support [38], [39].

The framework is built around two basic sets: a standard library for modeling supply chain processes, which allows companies to communicate and set up collaborative environments, and a set of standardized metrics, which allows supply chain managers to identify supply chain issues, benchmark, and establish supply chain management strategies.

SCOR proposes a hierarchical structure for describing supply chain processes. The primary building block of the SCOR model consists of five basic processes: plan, source, make, deliver, and return. In addition to these basic processes, there are three process types or categories, Enable, Planning, and Execute. The SCOR modeling approach starts with the assumption that any supply chain process can be represented as a combination of the five basic processes.



Figure 3 SCOR Model [41]

The five basic processes are referred to as process types [38]. The Plan process type describes the business activities associated to determining requirements to achieve targeted results. It includes five segments for planning supply chains: plan source, plan make, plan deliver and plan resource. The Source process type seizes business activities associated with ordering, delivery, receipt, and transfer of sourced products or services. Make process type aggregates the activities of adding value to the sourced products through different production processes. Deliver represents the business activities that transfer a product from the company to the client, and finally, the Return process type encompasses the activities associated with moving material from a customer back to the supplier.

Level 1 process types are compounded by level 2 processes. Level 2 processes are referred as process categories [38]. The purpose of these processes is to provide a standardized framework to further describe the process types. The plan process type is represented as P2, P3, P4, and P5 for plan source, plan make, plan deliver and plan return. The basic source, make, deliver, and return have variants like make to stock, make to order and engineer to order [40]. Thus, the combination of the process categories in thread diagrams describes the business model of the organization.

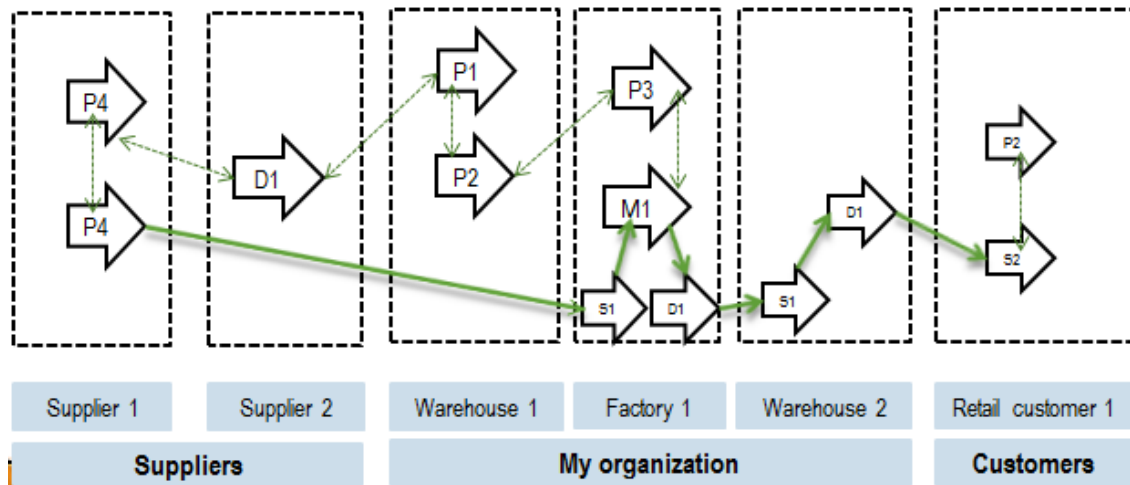


Figure 4 Thread Diagram [42]

Each of the level 2 process elements is further detailed in level 3 of the model. Level 3 processes describe the steps performed to execute the level 2 processes. The sequence in which these processes are executed influences the performance of the level 2 processes and the overall supply chain. Level 3 processes are referred to as activities or decompose processes [38]. Thus, if the modeling of the supply chain is not further described in level 4 than the level 3 processes are considered standard SCOR activities. Otherwise, the processes are considered decomposition processes of the level 2 process categories and are further detailed in business process models, workflow models, organizational charts etc. The SCOR vocabulary for level 3 processes is used to model supply chain workflows in a standardized way so that any SCOR expert could understand the business model of the organization.

5 Chapter 5: Development of a new SCOR Modeling Tool

In this chapter we describe a new Modeling Tool for supply chain operations. The tool is based on SCOR specifications, thus we will use the SCOR DSL and extend it to provide the features that, we believe, are mandatory for complete supply chain operations modeling.

5.1 Requirements for SCOR modeling tools

We think that our tool should address the following key management components:

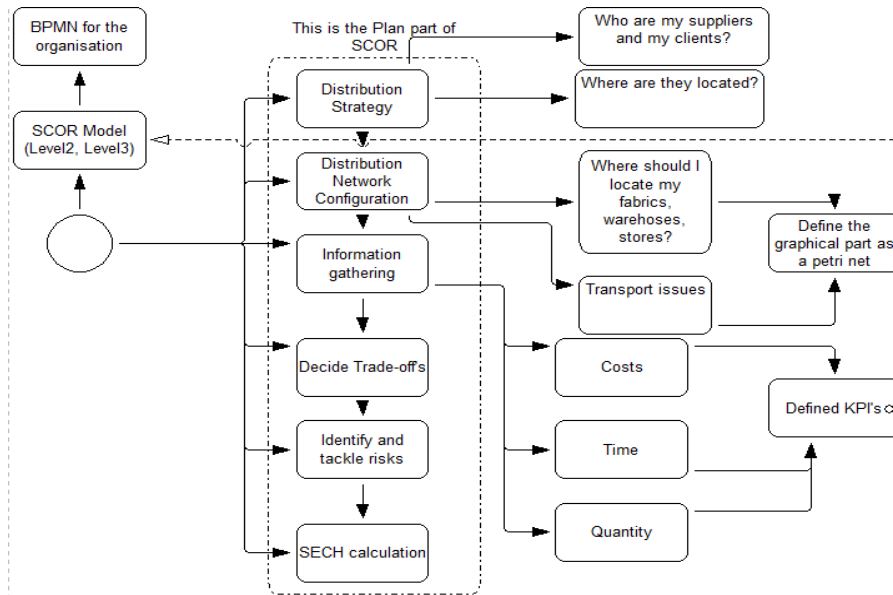


Figure 5 SCOR modeling tool requirements

Figure 8 provides an overview of the criteria described above, and the interdependence between them.

5.2 SCOR-based Modeling Language Conceptualization

We have started our journey of conceptualizing a new SCOR-based modeling tool having a top-down approach. The top-down approach is used when the model engineer starts at a high level, by analyzing the domain, then the required model stack, the model types and, at the end, the classes, relations and the syntax used to represent their instances in future models.

We start the SCOR-based language conceptualization by defining the contexts. When deployed in a meta-modeling platform, these contexts will become model types. As the model types contain notations and relations, we provide a conceptual example of how the model should look like. This approach allowed us to identify the notations and relations belonging to the model types.

5.2.1 Product model

Product model is a hierarchical representation of the decomposition of a product. In literature decomposition is typically known as the bill of materials (BOM). As business becomes more responsive to unique consumer tastes and derivative products grow to meet the unique

configurations, BOM management can become unmanageable. Advanced modeling techniques are necessary to cope with configurable products where changing a small part of a product can have multiple impacts on other product structure models [73].

The proposed model:

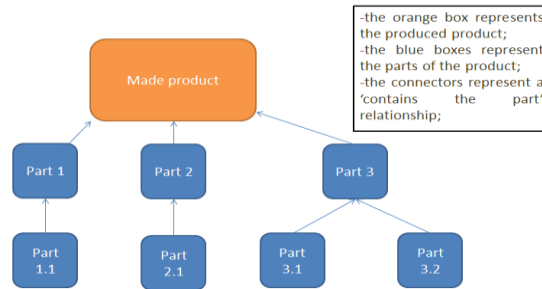


Figure 6 Product Model

For this model type we have the Product Model context, in which we define two notations (Product and Part), and one relation (Formed By). We used the Language Engineering Tool to describe this part of the language and the visual result based on ML² is listed in Figure 9 (ML² Product Model Overview):

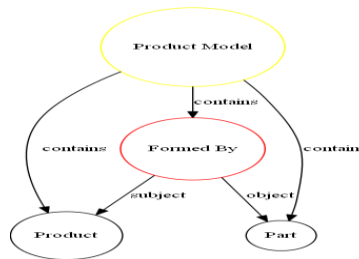


Figure 7 ML² Product Model Overview

5.2.2 Scope model

The Scope model is an overview of the environment in which an organization acts. It presents all the interactions and the material or information flows between the organization, its' partners, clients and competitors.

For this model type we have the Scope Model context, in which we define two notations (Swim-lane and Entity), and one relation (Material/Information Flow).

The proposed model:

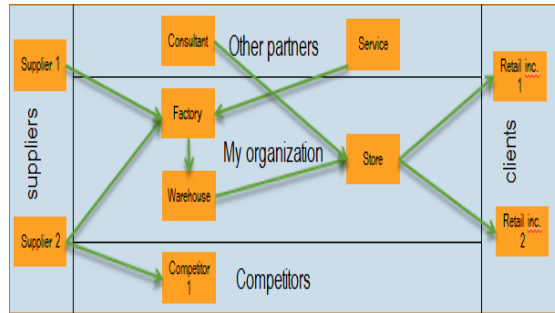


Figure 8 Scope Model

We used the Language Engineering Tool to describe this part of the language and the visual result based on ML^2 is listed in Figure 12 ML^2 Scope Model Overview.

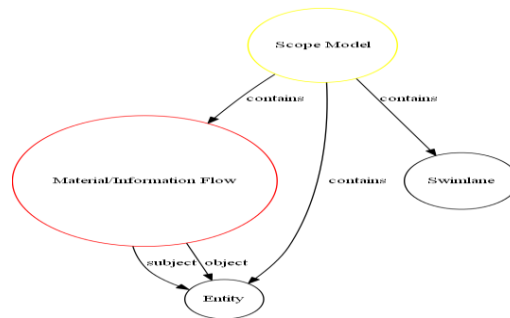


Figure 9 ML^2 Scope Model Overview

5.2.3 Sales region

The proposed model:

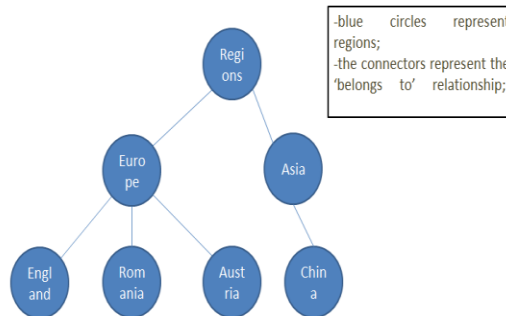


Figure 10 Sales Region Model

For this model type we have the Sales Region Model context, in which we define one notation (Region), and one relation (Detail Region). Definition of attributes and adverbs is subject of a future research. Thus, in this section we only define the language structure containing contexts, notations and relations.

We used the Language Engineering Tool to describe this part of the language and the visual result based on ML^2 is listed in Figure 14.

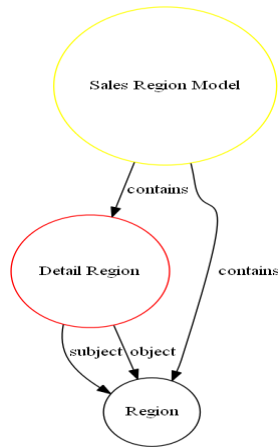


Figure 11 ML² Sales Region Model Overview

5.2.4 Supply regions

The proposed model:

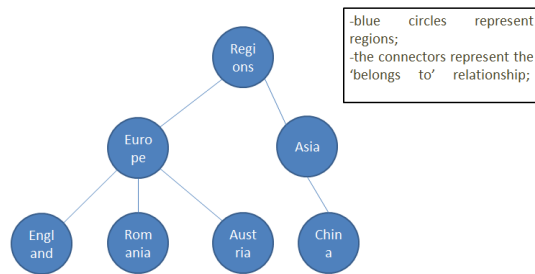


Figure 12 Supply Regions

For this model type we have the Supply Region Model context, in which we define one notation (Region), and one relation (Detail Region). Note that the notations and relations for this model type are the same as for the Sales Region Model, thus we can reuse them.

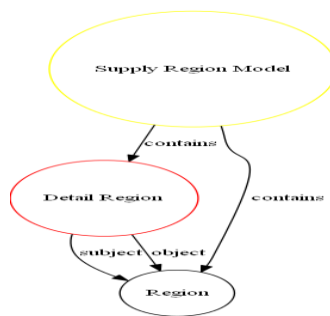


Figure 13 ML² Sales Region Model Overview

5.2.5 Geographical map

The proposed model:

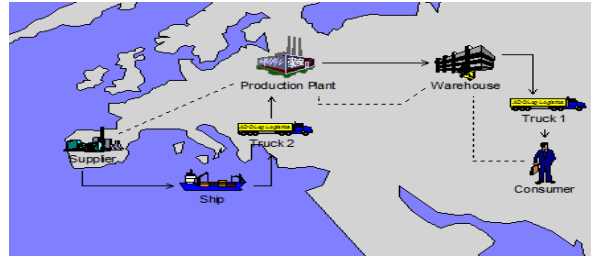


Figure 14 Geographical Model

For this model type we have the Geographical Model context, in which we define seven notations (Supplier, Plant, Warehouse, Consumer, Truck, Ship, and Map), and two relations (Information Flow, Material Flow).

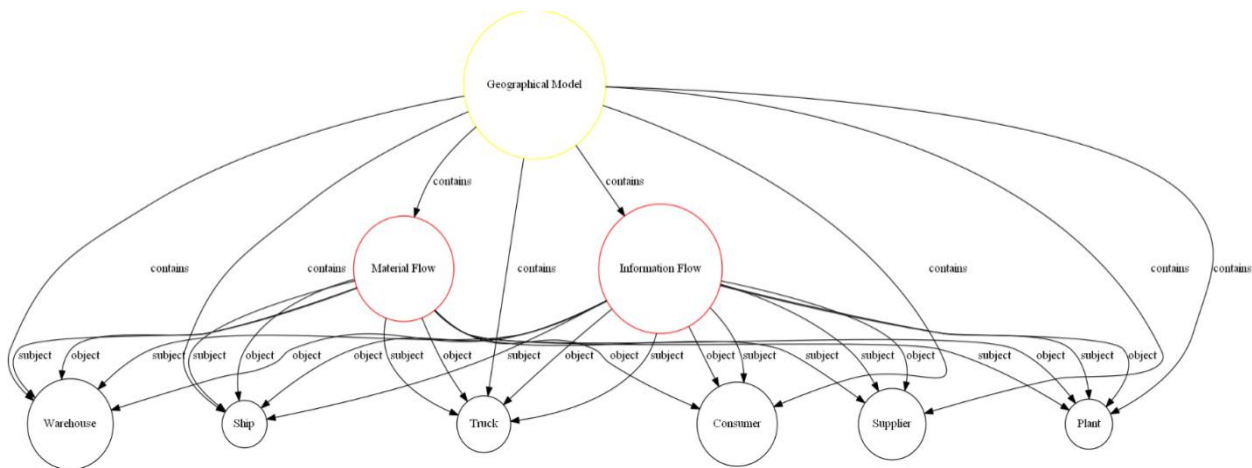


Figure 15 ML² Sales Region Model Overview

5.2.6 Extended thread diagram

Models:

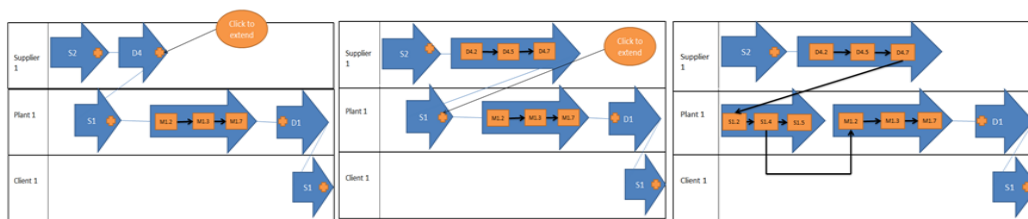


Figure 16 Extended Thread Diagram Models

For this model type we have the Extended Thread Diagram Model context, in which we define three notations (Swim-lane, Arrow, and SCOR Process), and two relations (Arrow Connector, Process Connector).

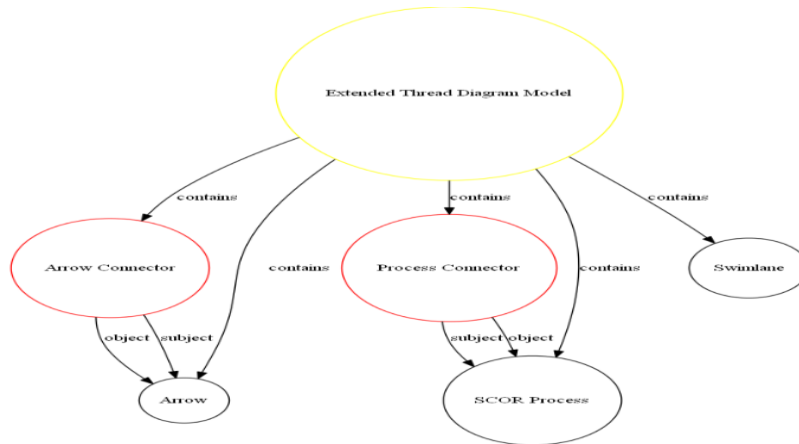


Figure 17 ML² Extended Thread Diagram Model Overview

5.2.7 Business Process model

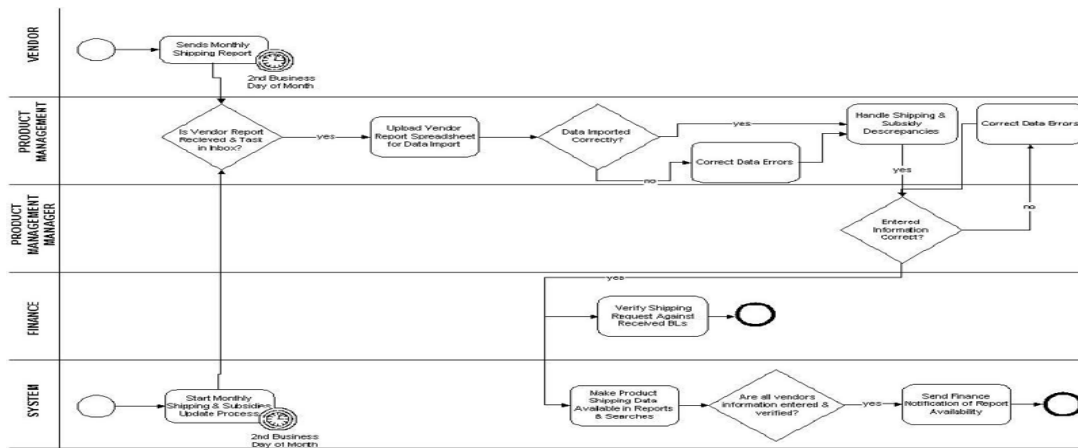


Figure 18 Business Process Model

For this model type we have the BP Model context, in which we define six notations (Swim-lane, Start, Activity, Timer, Decision, and Stop), and two relations (Activity Connector, Timer Connector).

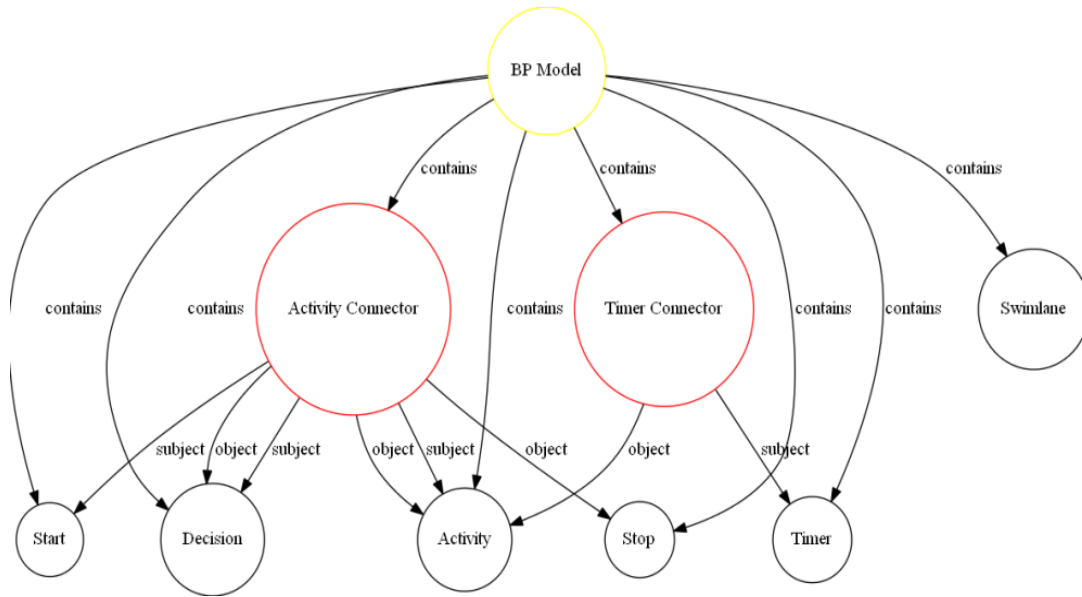


Figure 19 ML² Business Process Model Overview

5.2.8 Organizational model

The proposed model:

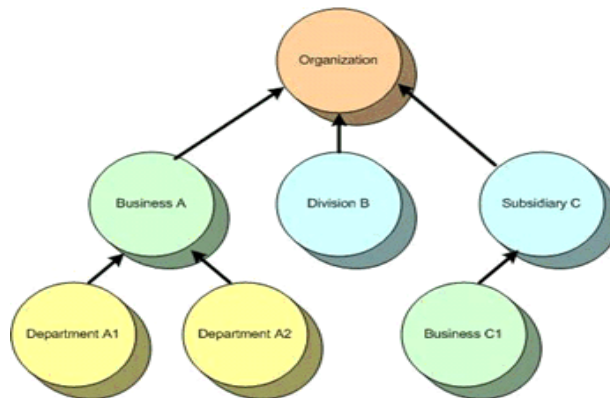


Figure 20 Organizational model

For this model type we have the Organizational Model context, in which we define one notation (Entity Type), and one relation (Decomposition).

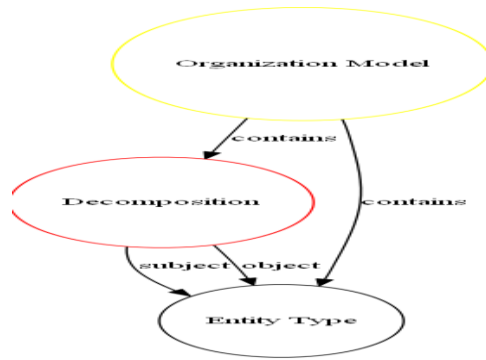


Figure 21 ML² Organization Model Overview

5.2.9 SCOR-based DSL Overview

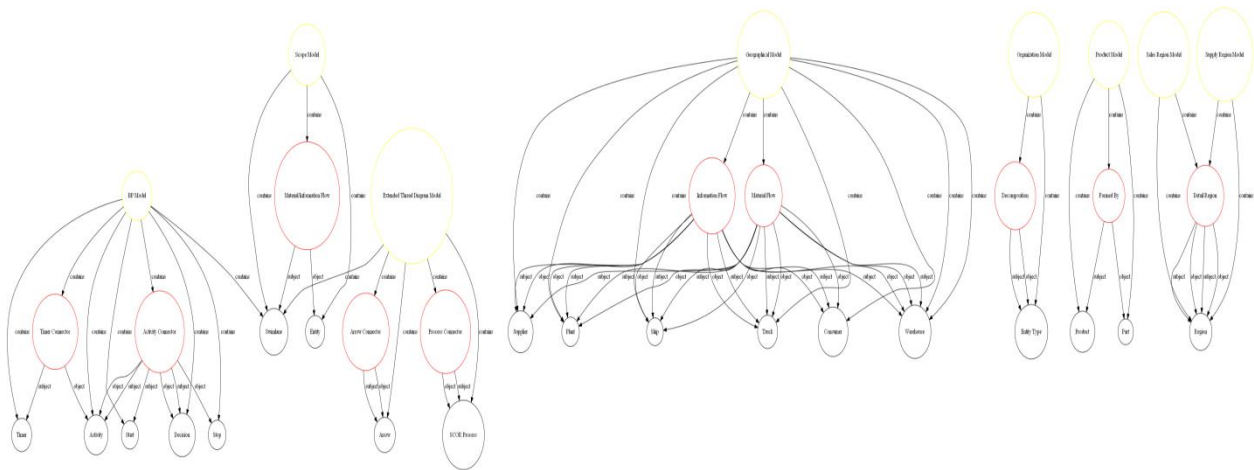


Figure 22 ML² SCOR-based DSL Overview

Figure 25 is the overview of the described models. It represents the visualization of the data set that can be, now, imported in a meta-modeling framework to build the new SCOR-based modeling tool.

Summary and Future Work

1 Contributions

The goal of this research was to develop a modeling method for supply chain management. But, along the research period, interesting results were also found. Thus, we can say that the contribution of the thesis is threefold.

Firstly, starting from an innovative definition of a language as a concept, and based on a simple hypothesis that states that a word is defined by a unique set of attributes; the authors construct a theory explaining every element of a language, the relations between these elements, and how statements are constructed.

Secondly, based on these definitions, authors describe the methodology of conceptualizing languages, and propose a tool meant to help language conceptualizers to describe new domain specific languages (DSLs). The tool allows its users to engineer, reengineer, analyze, and compare DSLs. The goal of this tool is to provide insight about languages. The outcome of the engineering module is a data set which can be easily transformed and imported in meta-modeling platforms to generate new modeling tools. The graph that is generated following the conceptualizing process describes the language in detail and it can be used to pass requirements to people in charge of implementing new DSLs or to validate the DSLs that were implemented on meta-language frameworks.

Lastly, the thesis presents a case study on conceptualizing a new modeling tool for supply chain process management, analysis, simulation, and optimization. The modeling tool will be constructed based on supply chain operations reference (SCOR) and the conceptualization phase will be conducted using our proposed Language Engineering tool.

Punctually, we can say that:

1. In **Chapter 1** we have managed to describe a language formally. To cut off the ambiguity between the readers and the authors we have emplaced definitions of concepts that were used among the thesis. Our definitions should be regarded as the first attempt to formally describe language as a concept and not as a specific linguistic system. The fundamental hypothesis of our approach was that all known languages have a common structure. Having a language formal definition as base, we were able to employ a language conceptualization methodology. The methodology could be used by language conceptualizers to deploy new languages. It implies constructing a graph of notations, relations, and contexts, which represents the structure of the new language.
2. In **Chapter 2** we have described a new modeling language for DSL modeling, which could support the language conceptualization methodology described in Chapter 1. The result, of this part of the research, was the ML^2 language. ML^2 resembles to MOF M3 but we enriched its syntax to address all the shortcomings we identified while analyzing MOF.
3. In **Chapter 3** we used ML^2 and the proposed language formal definition to develop a tool meant to provide its users all the functionalities required for Domain Specific Language

Engineering. The tool provides three modules: a reengineering module, a language description module, and a language analysis module.

4. In **Chapter 4** we have analyzed supply chains with the intention of conceptualizing a new modeling tool for supply chain operations. We have chosen to base our research on a very well-known DSL in the supply chains theory, which is SCOR. We have compared the SCOR framework with its main competitor VRM and were able to explain why we had chosen SCOR as base of our modeling tool.
5. In **Chapter 5** we have conceptualized a new modeling method for supply chain operations. We have conducted our work using the Engineering Tool presented in Chapter 3.

2 Future work

This thesis proposes innovative approach in three different research fields: economy, informatics, and linguistics. The described findings are the result of a research period that implied scaling up and down the abstraction levels which obliged the authors to study parts from all these domains. While studying around these fields ideas for future works have arisen:

1. **The economical field** was touched due to the main goal of the research which was to develop a modeling tool for supply chain processes. As we were able to conceptualize the needed DSL for such a tool and generated a data set that can be used as input for a meta-modeling framework we intend to analyze the existing meta-modeling platforms and decide which one shall we use to import our new modeling method? After we will import the data set described in Sub-section 5.6.9 we will study the platforms functionalities. The goal is to learn how we could add dynamic behavior to our modeling method. By dynamic behavior we understand analysis and simulation algorithms. Nevertheless, we intend to discover new algorithms that could analyze and simulate supply chain processes.
2. **Informatics** was studied as the authors propose a tool meant to help language conceptualizers to describe new domain specific languages (DSLs). The tool allows its users to engineer, reengineer, analyze, and compare DSLs. Our intention is to extend the Language Engineering tool to enable users to describe generic algorithms that could be imported in these meta-modeling frameworks so that, one who chooses to use our tool

shall not be forced to learn how a specific meta-modeling platform should be used to describe the algorithms.

3. **Linguistics** was studied because the authors had to provide an innovative definition of a language as a concept, which could explain how languages are conceptualized. As a future work we intend to study how could we use the language formal definition and its structure's graphical representation to respond to some NLP problems like, synonymy, similarity, text categorization, text validation, and even text translation.

Bibliography

1. **Pateman, Trevor.** <http://www.selectedworks.co.uk>.
<http://www.selectedworks.co.uk/whatisalanguage.html>. [Online] 2012. [Cited: 3 20, 2012.]
<http://www.selectedworks.co.uk/whatisalanguage.html>.
2. **Noe, Alva.** <http://www.npr.org/>. <http://www.npr.org/blogs/13.7/2011/04/04/135014573/what-is-a-language-when-easy-questions-demand-tough-answers>. [Online] 2012. [Cited: 2 20, 2012.]
3. *Syntactic Structures*. **Chomsky, Noam.** Hague : the Hague: Motion, 1957.
4. *In Search of a Basic Principle for Model Driven Engineering*. **Bézivin, J.** 21-24, s.l. : Cepis UPGRADE, 2004, Vol. The European Journal for the Informatics Professional V(2).
5. *A Framework to Formalise the MDE Foundations*. **Thirioux, Xavier, et al.** Zurich : Suisse : International Workshop on Towers of Models, 2007, Vol. TOWERS 2007.
6. **Meta-Object-Facility.** http://en.wikipedia.org/wiki/Meta-Object_Facility.
<http://en.wikipedia.org>. [Online] 2012. [Cited: January 16, 2012.]
http://en.wikipedia.org/wiki/Meta-Object_Facility.
7. *"Heuristics-Based Query Processing for Large RDF Graphs Using Cloud Computing"*.
Husain M, McGlothlin J., Masud M., Khan L., Thuraisingham B.M. s.l. : Knowledge and Data Engineering, IEEE Transactions, vol. 23, pages 1312-1327,, 2011.
8. *"Heuristics-Based Query Processing for Large RDF Graphs Using Cloud Computing"*.
Husain, M., et al. s.l. : Knowledge and Data Engineering, IEEE Transactions, 2011, Vols. vol. 23, pages 1312-1327.
9. *Model-driven Language Engineering: the ASMETA case study*. **Gargantini, Angelo, Riccobene, Elvinia and Scandurra, P.** s.l. : Software Engineering Advances, 2008. The Third International Conference on Software Engineering Advances.
10. *Formal Grammars and Languages*. **Tao Jiang, Ming Li.** s.l. : Chapman & Hall/CRC ©2010, 2010. Algorithms and theory of computation handbook.
11. **umbc.** [http://www.csee.umbc.edu/portal/help/theory/lang_def.shtml]. www.csee.umbc.edu. [Online] umbc. [Cited: 9 22, 2012.]
[http://www.csee.umbc.edu/portal/help/theory/lang_def.shtml].
12. *Formal language theory for natural language processing*. **Wintner, Shuly.** Philadelphia : ACL'02 Workshop, 2002. ACL'02 Workshop.

13. *"Model-based DSL Frameworks"*. **I. Kurtev, J. B'ezivin, F. Jouault and P. Valduriez**. s.l. : Companion to the 21st Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA, 2006.
14. *Exploratory Testing*. **Kaner, Cem**. Orlando, FL : Florida Institute of Technology, 2006. Quality Assurance Institute Worldwide Annual Software Testing Conference.
15. **BOC**. www.openmodels.at . *ADOxx platform description*. [Online] BOC GmbH. [Cited: 2 12, 2012.]
16. *Towards a comparative analysis of meta-meta-models*. **Heiko Kern, Alex Hummel, Stephane Kuhne**. New York : ACM New York, 2011. SPLASH '11 Workshops Proceedings of the compilation of the co-located workshops on DSM'11, TMC'11, AGERE!'11, AOOPEs'11, NEAT'11, & VMIL'11.
17. *The Generic Modeling Environment*. **A. Ledeczki, M. Maroti, A. Bakay, G. Karsai, J. Garrett, C. Thomason, G. Nordstrom, J. Sprinkle, P. Volgyesi**. s.l. : IEEE, 2001. IEEE International Workshop on Intelligent Signal Processing.
18. **Eclipse**. <http://www.eclipse.org/modeling/emf/>. <http://www.eclipse.org/modeling/emf/>. [Online] [Cited: 3 12, 2012.] <http://www.eclipse.org/modeling/emf/>.
19. **Meta-Object-Facility**. http://en.wikipedia.org/wiki/Meta-Object_Facility. www.wikipedia.org. [Online] 2012. [Cited: 2 20, 2012.] http://en.wikipedia.org/wiki/Meta-Object_Facility.
20. *i* on ADOxx®: A Case Study*. **Margit Schwab, Dimitris Karagiannis, Alexander Bergmayr**. Hammamet, Tunisia : s.n., 2010. iStar2010 - Proceedings of the 4th International i* Workshop.
21. **OMG**. *OMG, UML 2.0 Infrastructure Specification*, www.omg.org. s.l. : OMG, 2012.
22. *"Abstract Syntax from Concrete Syntax"*. **Wile, David S**. Boston, MA, USA; : ICSE 1997, 1997.
23. *"A metamodel for the notation of graphical modeling languages"* . **Xiao He, Zhiyi Ma, Weizhong Shao, Ge Li**. s.l. : 31st Annual International Computer Software and Applications Conference (COMPSAC), 2007;.
24. **Babbie, Earl R**. *The Practice of Social Research*. s.l. : Cengage Learning, 2010. ISBN 0495598410, p. 14-18.
25. *"ARIS: Business Process Modeling"*. **Scheer, A.-W**. s.l. : Springer, 2000.

26. *"Three Factors in Language Design"*. **Chomsky, Noam**. s.l. : Linguistics Inquiry 36, no.1, 2005.
27. *"The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM Conference"*. **Backus, J.W.** Zurich : Proceedings of the International Conference on Information Processing, UNESCO. pp. 125–132, 1959;.
28. **Heiko Kern, Alex Hummel, Stephane Kuhne**. "Towards a comparative analysis of meta-meta-models". <http://www.dsmforum.org/events/DSM11/Papers/kern.pdf>. [Online] [Cited: 4 15, 2012.] <http://www.dsmforum.org/events/DSM11/Papers/kern.pdf>.
29. *"Domain-Specific Modeling: Enabling Full Code Generation"*. **S. Kelly, J.-P. Tolvanen**. s.l. : John Wiley& Son, Inc., 2008.
30. *"Meaning as use: a functional view of semantics and pragmatics"*. **Mwihaki, Alice**. s.l. : SW AHILI Forum 11, pg. 127-139, 2004;.
31. **S. Cook, G. Jones, S. Kent, and A. C. Wills**. *"Domain Specific Development with Visual Studio DSL Tools (Microsoft .Net Development)"*. s.l. : Addison-Wesley Longman, 2007.
32. *UIMA: an architectural approach to unstructured information processing in the corporate research environment*. **David Ferrucci, Adam Lally**. DAVID FERRUCCI and ADAM LALLY (2004). UIMA: an architectural approach to unstructured information processing in the corporate research enviro10 , 2004, Vol. Natural Language Processing. pp 327-348. doi:10.1017/S1351324904003523. .
33. **graphviz**. <http://www.graphviz.org/>. <http://www.graphviz.org/>. [Online] [Cited: 09 22, 2011.] <http://www.graphviz.org/>.
34. **Glinz, Martin**. *VLML Presentation*. Vienna : s.n., 2011.
35. *On the Theoretical Foundation of Meta-Modelling in Graphically Extended BNF and First Order Logic*. **Zhu, Hong**. Taipei, Taiwan : Theoretical Aspects of Software Engineering (TASE), pages 95-104, 2010. 4th IEEE Symposium.
36. *The Design of a Conceptual Framework and Technical Infrastructure for Model Management Language Engineering*. **Richard F. Paige, Dimitrios S. Kolovos, Louis M. Rose, Nicholas Drivalos, Fiona A.C. Polack**. s.l. : 14th IEEE International Conference on Engineering of Complex Computer Systems, 2009.

37. *Model-Driven Engineering of a General Policy Modeling Language*. **Nima Kaviani, Dragan Gasevic, Milan Milanovic, Marek Hatala, Bardia Mohabbati**. s.l. : IEEE Workshop on Policies for Distributed Systems and Networks, 2008.
38. **SCC**. *Supply Chain Operations Reference Model Version 10*. s.l. : Supply Chain Council, 2011.
39. *Systemic Assessment of SCOR for Modeling Supply Chains*. **Kasi, Vijay**. 2005. Proceedings of the 36th Hawaii International Conference on System Sciences.
40. *An Introduction to the Supply Chain Council's SCOR Methodology*. **Hermon, P.** 2003. Business Process Trends.
41. **SCC**. <http://scor-software-model-framework.softsia.com/>. <http://scor-software-model-framework.softsia.com/>. [Online] [Cited: 6 12, 2012.]
42. **Supply Chain Council**. <http://supply-chain.org/f/SCOR%2090%20Overview%20Booklet.pdf>. <http://supply-chain.org>. [Online] 2009. [Cited: 9 12, 2011.]
43. *Supply Chain Management*. **Zigiaris, Sotiris**. 2000. BPR Hellas SA .
44. **Supply Chain Council**. www.supply-chain.org. www.supply-chain.org. [Online] 2009. [Cited: 7 27, 2012.]
45. *HP's global supply chain optimization*. **Becker, Mark**. Singapore : s.n., 2011. presentation, http://supply-chain.org/f/M_Bakker_SCC_SupplyChainWorld_v1.pdf.
46. **Motorrad, BMW**. <http://supply-chain.org/f/12%20-%20Stadelhofer%20-%20Innovative%20Supplier%20Management%20Systems%20at%20BMW%20Motorcycles.pdf>. www.supply-chain.org. [Online] Supply Chain Council, 10 25, 2011. [Cited: 10 22, 2012.]
47. **Brun, Stein Erland**. "Supplying consumables to offshore oil and gas plants", http://supply-chain.org/f/STATOIL_SCC.pdf. [Online] 10 11-12, 2003. [Cited: 10 25, 2012.]
48. *High Performance through Process Excellence*. **Kirchmer, M.** 2011, Springer-Verlag Berlin Heidelberg.
49. *Business process reference models: Survey and classification*. **Fettke, P., Loos, P. and Zwicker, J.** 2006, BPM 2005 Workshops, Bussler at al. (Eds.), LNCS 3812, Springer, pp. pp. 469-483.
50. **Blog**. www.supplychainadvice.wordpress.com. [Online] 01 25, 2010. [Cited: 11 20, 2010.] supplychainadvice.wordpress.com/2010/01/25/scor-vs-the-value-reference-model-frm/.

51. *Value Chains Versus Supply Chains*. **Feller A., Shunk D, Callarman T.** s.l. : <http://www.ceibs.edu/knowledge/papers/images/20060317/2847.pdf>, 2006. BPTrends.
52. *Business process reference models: Survey and classification*. **Fettke P., Loos P., Zwicker J.** s.l. : Bussler at al. (Eds.), LNCS 3812, Springer, 2006. BPW 2005 Workshops. pp. 469-483.
53. **Bolstoff P., Rosenbaum R.** *Supply Chain Excellence -3rd Edition*. s.l. : AMACOM, 2011.
54. **Value-Chain Group.** *Business process library*. <http://www.value-chain.org/business-process-library/public/>.
55. *Enterprise and process architecture patterns*. **Barros O, Julio C.** 2011, Business Process Management Journal, Vol. 17, No. 4, pp. 598-618.
56. **Value Chain Group.** *Value-Chain Group's and Education and Training* . [http://www.value-chain.org/education and training/](http://www.value-chain.org/education%20and%20training/).
57. **Harmon, P.** *Business Process Change, Second Edition: A Guide for Business Managers and BPM and Six Sigma Professionals*. s.l. : The MK/OMG Press, 2007.
58. **value chain org.** www.value-chain.org. [Online] [Cited: 02 08, 2012.] <http://www.value-chain.org/businee-process-library/public/>.
59. *IEEE Standard for Software Reviews*. s.l. : IEEE Std. 1028-1997, 1994.
60. *APICS Dictionary*. **James F. Cox, John H., Jr. Blackstone, James F. III Cox.** 2005. ISBN-10: 1558221913.
61. **Zigiaris, Sotiris.** *Supply Chain Management*. s.l. : BPR Hellas SA, 2000.
62. *Determining Criteria for Software Components: Lessons Learned*. **Juan Pablo Carvalho, Xavier Franch, Carmen Quer.** s.l. : IEEE Software, 2007.
63. **Wikipedia Org.** http://en.wikipedia.org/wiki/ISO/IEC_9126. [Online] [Cited: 10 22, 2012.] http://en.wikipedia.org/wiki/ISO/IEC_9126.
64. **ISO/IEC.** *ISO/IEC 9126-1, White-Paper First edition*. s.l. : ISO/IEC, 2001-06-15.
65. *Modeling Software Measuring Data*. **B. A. Kitchenham, R. T. Hughes, S.G. Linkman.** 9, pp.:788-804, s.l. : IEEE Trans Software Eng., 2001, Vol. 27.
66. **Tavaf, Saeedeh Jadid.** *Quality Evaluation in transformation of event logs into visual representations*. Gothenburg, Sweden : Master Thesis in Software Engineering and Management,, 2009. ISSN 1651-4769.
67. **Wal-Mart.** *Wal-Mart's Sustainability Index and Supply Chain Green Standards*. 2008.
68. **IDS Scheer.** *ARIS Platform Products Version: 7.02*.

69. **TIBCO**. http://developer.tibco.com/business_studio/default.jsp. [Online] [Cited: 10 24, 2012.] http://developer.tibco.com/business_studio/default.jsp.
70. **Metastorm ProVision**. <http://www.metastorm.com>. [Online] Product Overview Metastorm ProVision. [Cited: 11 22, 2011.]
71. **Nimbus**. *Business Process Management within the Value Chain, Nimbus whitepaper*; .
72. **ProcessWizard**. <http://supply-chain.org/scor-software-processwizard>. [Online] [Cited: 10 23, 2012.] <http://supply-chain.org/scor-software-processwizard>.
73. **Hvam, L.** *A procedure for building product models. Robotics and Computer-Integrated Manufacturing*. 1999. pp. 77-87.
74. *Abstract Syntax from Concrete Syntax*. **Wile, David S.** Boston, MA, USA; ; ICSE, 1997.
75. "Model-Driven Engineering". **Schmidt, D.C.** s.l. : IEEE Computer 39 , 2006, Vol. 2.
76. **Pateman, Trevor**. *Language in Mind and Language in Society*. 1987.
77. **OMG**. www.omg.org UML 2.0 Infrastructure Specification. www.omg.org. [Online] OMG. [Cited: 3 17, 2012.] www.omg.org;
78. —. *OMG, UML 2.0 OCL Specification, www.omg.org*; . s.l. : OMG, 2012.
79. *Elements of General Linguistics*. **Martinet, André**. London : (1960). Elements of General Linguistics. Tr. Elisabeth Palmer Studies in General Linguistics, 1960, Vol. 1.
80. **Lyons, John**. *Language and Linguistics*. s.l. : Cambridge University Press, 1981.
81. **Lockerby, Patrick**. http://www.science20.com/chatter_box/blog/what_language. <http://www.science20.com/>. [Online] 5 19, 2009. [Cited: 06 19, 2012.] http://www.science20.com/chatter_box/blog/what_language.
82. **Lightfoot, David**. Introduction by David W. Lightfoot. [book auth.] Noam Chomsky. *Semantic Structures*. Berlin : Walter de Gruyter GmbH & Co. KG, 1957.
83. *Wikipedia-based Semantic Interpretation*. **Gabrilovich, Evgeniy and Shaul, Markovitch**. s.l. : urnal of Arti- cial Intelligence Research, 2009, Vol. 34. 0738-4602.
84. *WordNet: An Electronic Lexical Database*. . **Fellbaum, C.** Cambridge : MIT Press, 1998.
85. *Metamodelling Platforms* . **Dimitris Karagiannis, Harald Kühn**. Dexa, Aix-en-Provence, France : In Bauknecht, K., Min Tjoa, A., Quirchmayer, G. (Eds.): Proceedings of the Third International Conference EC-Web 2002, LNCS 2455, Springer, Berlin, 2002.

86. **dictionary.reference.com**. <http://dictionary.reference.com/browse/language>.
<http://dictionary.reference.com>. [Online] [Cited: 06 19, 2012.]
<http://dictionary.reference.com/browse/language>.
87. **Dictionaries, Oxford**. <http://oxforddictionaries.com/definition/language>.
<http://oxforddictionaries.com/>. [Online] 06 19, 2012. [Cited: 06 19, 2012.]
<http://oxforddictionaries.com/definition/language>.
88. *Indexing by latent semantic analysis*. **Deerwester, S., et al.** 6, s.l. : Journal of the American Society for Information Science, 1990, Vol. 41.
89. *Lingvistica*. **Cohn, Anthony**. 1999, xxx, pp. 13-29.
90. **Bruce, Bower**. Talking back in time; prehistoric origins of language attract new data and debate - language evolution. *Science News on Bnet (Technology Industry)*. *CBS Interactive News Service*. . 29 September, 2000.
91. *Modern Information Retrieval*. **Baeza-Yates, R. and Ribeiro-Neto, B.** New York : Addison Wesley, 1999.
92. **ADONIS**. <http://www.adonis-community.com/>. <http://www.adonis-community.com/>.
[Online] BOC, 2011. [Cited: 11 22, 2011.] <http://www.adonis-community.com/>.
93. *Joint Use of SCOR and VRM*. **Costin, Răzvan Aurelian, Kirikova, Marite and Buchmann, Robert**. 2012, in Perspectives in business informatics research : 11th international conference, BIR 2012 Proceedings, Nizhny Novgorod, Russia, ed. Springer, ISBN: 3-642-33280-3,.
94. **value-chain.org**. www.value-chain.org/. [Online] 10 08, 2012. [Cited: 10 08, 2012.]
www.value-chain.org/bptf/buildingblocks/vrm/.
95. *Value Chains, Value Streams, Value Nets, and Value Delivery Chains*. **W., Brown G.** s.l. : <http://www.bptrends.com/publicationfiles/FOUR%2004-009-ART-Value%20Chains-Brown.pdf>, 2009. BPTrends;

Declarations:

96. **Costin Aurelian Răzvan**, Țolea Eniko. „E-collaboration - The new economic world”, in *Journal of Information Systems & Operations Management*, Vol. 4, Nr. 1, pg. 142-149, ISSN 1843-4711, CNCSIS B+, Mai 2010;
97. **Costin Aurelian Răzvan**, „E-collaborative system for management of virtual organizations”, in *International Conference Managerial Challenges of the Contemporary Society Proceedings*, Cluj-Napoca, 4-5 June, 2010;

98. **Costin Aurelian Răzvan**, Țolea Eniko, „Deployment features for managing virtual organizations” in *Informatica Economica Journal*, September 2010;
99. **Costin Aurelian Răzvan**, Țolea Eniko, „Ontology for an E-learning model”. In *ICVL 2010: Proceedings of The 5th International Conference on Virtual Learning*, ISSN 1844-8933, Nr. 2689/2010, Editura Universității din București, pg. 135;
100. **Costin Aurelian Răzvan**, Rusu Lucia, Simona Kleinhempel, Sergiu Jecan, „A Collaborative Model For Virtual Enterprise” in *Journal of Information Systems & Operations Management*, Vol. 4, Nr. 2, pg. 20-32, ISSN 1843-4711, CNCSIS B+, December 2010;
101. **Costin Aurelian Răzvan**, Lucia Rusu, Marius Podean, Szasz Szabolcs, „A Hierarchical Model For Medical Registrations” in *Journal of Information Systems & Operations Management*, Vol. 4, Nr. 2, pg. 20-32, ISSN 1843-4711, CNCSIS B+, December 2010;
102. **Costin Aurelian Răzvan**, Țolea Eniko, „The Design of a Distributed Database for Doctoral Studies Management”. In *Informatica Economica Journal*, Vol. 14, No. 4/2010, pg. 139-146, ISSN 1453-1305, CNCSIS B+, December 2010;
103. **Costin Aurelian Răzvan**, Țolea Eniko, „Software Development Business Moving towards a Unified Collaborative System in IT. SaaS may be the New Orientation.”. *Analele Universității din Oradea*. TOM XIX nr. 2, pg. 398-404, ISSN 1582-5450, CNCSIS B+, December 2010;
104. **Costin Aurelian Răzvan**, Țolea Eniko, Marius Podean, „Integrating SaaS Modules Using XML Technologies for Collaborative Project”, in *CEE Symposium 2011 Proceedings*, 28-30 April, 2011;
105. **Costin Aurelian Răzvan**, Jecan Sergiu, Țăranu Mirela, „Integrated Management System in College Admission”, in *IECS Proceedings*, nr. 18, Thomson Reuters-ISI Proceedings, 19 Mai 2011;
106. **Costin Aurelian Răzvan**, Marius Podean, Dan Bența, „On Supporting Creative Interaction in Collaborative Systems: A Content Oriented Approach”, in *IEEE 13th Conference on Commerce and Enterprise Computing Proceedings*, ISBN: 978-0-7695-4535-6, <http://doi.ieeecomputersociety.org/10.1109/CEC.2011.65>, pg. 400-406, October 2011;
107. **Costin Aurelian Răzvan**, Lucia Rusu, Țolea Eniko, Sergiu Jecan, „Integrated Platform for Virtual Enterprise”, in *TELFOR proceedings*, ISBN:9781457714986, Publisher: Institute of Electrical and Electronics Engineers (IEEE), November 2011;

108. **Costin Aurelian Răzvan**, Țolea Eniko, „Collaborative E-Learning Model”, in Proceedings of the 6th International Conference on E-Learning ICVL, pg. 248-253, October 2011;
109. **Costin Aurelian Răzvan**, Lucia Rusu, Simona Marțiș, Marius Podea, „An Ehr Document Model of Chronic Patients” in Journal of Information Systems & Operations Management, ISSN: 1843-4711, nr. 5, pg. 264-275, December 2011;
109. **Costin Aurelian Răzvan**, Jecan Sergiu, Maria Iulia Laba, „Document Management in Context of Collaborative Systems”, in Journal of Information Systems & Operations Management, ISSN: 1843-4711, nr. 5, pg. 308-317, December 2011;
110. **Costin Aurelian Răzvan**, „Using blackboard-like software for coupling SaaS”, in Procedia – Computer Science Journal, ISSN: 1877-0509, Mai 2012;
111. **Costin Aurelian Răzvan**, Marite Kirikova, Robert Buchmann, „Joint Use of SCOR and VRM”, in Perspectives in business informatics research : 11th international conference, BIR 2012 Proceedings, Nizhny Novgorod, Russia, ed. Springer, ISBN: 3-642-33280-3, 978-3-642-33280-7, September 2012;
112. **Costin Aurelian Răzvan**, „Dsl Reengineering Tool Development”, in The 12th International Conference on Informatics in Economy Proceedings, Bucharest, Romania, April 2013;
113. **Costin Aurelian Răzvan**, „New Modeling Language for Business DSLs”, in AWERProcedia Information Technology & Computer Science, accepted for publication, Antalya, Turkey, April 2013.