A Mechanism for Representing N-Dimensional Software Process Models in One-Dimensional Documents

Muhammad Saqib Saeed
This thesis is submitted to the School of Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author:
Muhammad Saqib Saeed
E-mail: msaqibsaeed@hotmail.com

External advisor:
Ove Armbrust
Department of Process and Measurement
Fraunhofer-Institute for Experimental Software Engineering (IESE)
Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
Phone: +49 631 6800-2259   E-mail: Ove.Armbrust@iese.fraunhofer.de

University advisor:
Dr. Mikael Svahnberg
School of Engineering

Blekinge Institute of Technology
PO Box 520
SE – 372 25 Ronneby
SWEDEN

Internet : www.bth.se/tek
Phone : +46 457 38 50 00
Fax : + 46 457 271 25
Abstract

Current software process modeling tools lack the capabilities of generating word processing documents that can represent model semantics in a computer process-able and human understandable way. This results into inefficient use of word processors for editing and reviewing a model’s textual data.

In an attempt to resolve this problem, this thesis presents an approach for representing software process models in word processing documents. The development of the approach is based on a set of issues that can hinder the generation of human understandable and computer process-able documents from software process models.

The approach is validated through its implementation for a software process modeling tool. The implementation allows for the generation of word processing documents from software process models and their re-import into the process modeling tool.

Keywords: Software process modeling, semantic documents.
Table of contents

1. INTRODUCTION .......................................................................................................................... 1
  1.1 MOTIVATIONS FOR THE THESIS............................................................................................. 2
  1.2 THESIS CONTRIBUTIONS ......................................................................................................... 4
  1.3 THESIS OUTLINE ..................................................................................................................... 4

2. RESEARCH APPROACH ............................................................................................................... 5
  2.1 RESEARCH QUESTIONS ............................................................................................................. 5
  2.2 RESEARCH STEPS ....................................................................................................................... 5
    2.2.1 Analysis of the problem ......................................................................................................... 6
    2.2.2 Survey of already available approaches .............................................................................. 6
    2.2.3 Analysis of constraints on the solution .................................................................................. 6
    2.2.4 Development of the solution ................................................................................................ 7
    2.2.5 Static validation ..................................................................................................................... 7
    2.2.6 Dynamic validation .............................................................................................................. 7

3. RELATIONSHIPS ......................................................................................................................... 9
  3.1 RELATIONSHIP TYPE AND INSTANCE .................................................................................... 9
  3.2 DEGREE OF A RELATIONSHIP .................................................................................................. 10
  3.3 CARDINALITY CONSTRAINT ...................................................................................................... 10
  3.4 PARTICIPATION CONSTRAINT ................................................................................................ 11
  3.5 CATEGORIES OF RELATIONSHIPS ......................................................................................... 12
    3.5.1 Hierarchical relationships ................................................................................................... 12
    3.5.2 Non-hierarchical relationships ........................................................................................... 14

4. RELATIONSHIP REPRESENTATION ISSUES .............................................................................. 15
  4.1 REPRESENTING ONE-TO-ONE RELATIONSHIPS .................................................................... 15
  4.2 REPRESENTING ONE-TO-MANY RELATIONSHIPS ................................................................ 16
  4.3 REPRESENTING MANY-TO-MANY RELATIONSHIPS ............................................................... 16
  4.4 REPRESENTING MULTIPLE TYPES OF RELATIONSHIPS ...................................................... 17
  4.5 REPRESENTING CONDITIONS .................................................................................................. 18
  4.6 REDUNDANCY .......................................................................................................................... 19
  4.7 COMPUTER PROCESSABILITY ............................................................................................... 20

5. BASICS AND CONSTRAINTS OF THE SOLUTION .................................................................... 21
  5.1 THREE-STEP-METHODOLOGY .............................................................................................. 21
    5.1.1 Examples of three-step-methodology .................................................................................. 22
  5.2 ADVANTAGES OF USING THREE-STEP-METHODOLOGY .................................................... 22
  5.3 REQUIREMENTS FOR USING THREE-STEP-METHODOLOGY .............................................. 23
  5.4 SELECTION AND EVALUATION OF A WORD PROCESSOR .................................................... 24
  5.5 CONSTRAINTS OF THE APPROACH FOR REPRESENTING MODELS IN DOCUMENTS .......... 25

6. REPRESENTING SPMS IN MSWORD DOCUMENTS .................................................................. 26
  6.1 CREATE META-MODEL ............................................................................................................. 27
    6.1.1 Constraints on many-to-many hierarchical relationships .................................................... 28
  6.2 BUILD RULES ........................................................................................................................... 29
    6.2.1 Create visual presentation rules ............................................................................................ 30
    6.2.2 Generate Schema ................................................................................................................ 32
  6.3 ANNOTATE DOCUMENT ......................................................................................................... 35
    6.3.1 Interleaving of custom schema tags and formatting tags ..................................................... 36

7. STATIC VALIDATION .................................................................................................................. 38
  7.1 RESTRICTIONS TO USE APPENDIX I RULES ...................................................................... 38
  7.2 MODEL SECTION IS NEEDLESS ............................................................................................. 39
  7.3 SUB-ENTITIES SECTION IS NEEDLESS ................................................................................... 39

8. DYNAMIC VALIDATION .............................................................................................................. 40
8.1 VALIDATION OF PRACTICABILITY OF THE APPROACH ........................................................... 41
  8.1.1 Limited space for representing multiple levels of sub-entities........................................ 41
  8.1.2 Lack of support for formatted data representation......................................................... 41
  8.1.3 Limitation to use textual-data outside of custom-schema tags ....................................... 41
8.2 VALIDATION OF USE OF APPROACH FOR GENERATING AND IMPORTING WORDML DOCUMENTS............................................................................................................................... 42
8.3 VALIDATION OF USEFULNESS OF THE APPROACH ................................................................. 42
  8.3.1 Representation of properties........................................................................................... 43
  8.3.2 Representation of hierarchical relationships................................................................. 43
  8.3.3 Representation of associations and flow-conditions.................................................... 43
  8.3.4 Navigation through the model ...................................................................................... 44
  8.3.5 Model clarity................................................................................................................. 44
  8.3.6 Solution as a whole...................................................................................................... 44
  8.3.7 Limitations of the solution ........................................................................................... 44
9. CONCLUSION .......................................................................................................................... 46
  9.1 ANSWERS TO THE RESEARCH QUESTIONS AND RESEARCH CONTRIBUTIONS............. 46
  9.2 LESSONS LEARNED ............................................................................................................ 48
    9.2.1 Building human understandable semantics is challenging.......................................... 48
    9.2.2 Hierarchical relationships are easy to represent......................................................... 48
    9.2.3 Existence of three steps for building semantic documents........................................ 48
10. FUTURE WORK.................................................................................................................... 49
  10.1 RELEASE OF NEXT VERSION OF THE APPROACH.......................................................... 49
  10.2 VALIDATION OF COMPLETE REVIEW PROCESS ........................................................ 49
  10.3 RELEASE OF THE IMPLEMENTED SOLUTION FOR INDUSTRY USE ............................ 49
REFERENCES .................................................................................................................................. 50
APPENDIX I - VISUAL PRESENTATION RULES........................................................................... 53
APPENDIX II – QUESTIONNAIRE.............................................................................................. 59
1. Introduction

Since the advent of software industry, software organizations have been facing problems of missed schedules, over expenditures, burnt-out engineers, and poor-quality products [1]. Nevertheless, software continues to play a vital role in our daily life. From safety and security critical systems to normal home appliances, almost every product or service has started exploiting software capabilities. This widespread use has resulted in more complex software systems, thus increasing both the number and the severity of above problems.

This has led researchers and practitioners to find ways for improving software quality and development productivity. In this context, one important research direction is the study of processes for managing, developing and maintaining software systems [2]. By definition, “a software process is a set of activities, methods, practices, and transformations that people use to develop and maintain software and associated products” [3]. Software processes became a distinct research area in the 1980s [4] and since then researchers explored many sub-areas of the field.

One of the areas forming the core of software process research is software process modeling [4]. A software process model (SPM) helps describing a software process and while serving this purpose it aims for the fulfillment of three primary objectives [4]:

1. Understanding the process
2. Improving the process
3. Enacting the process

In addition Alfonso et al [4] also presented a supporting objective that is common in all three primary objectives. It is called analysis of the process model. The analysis of the process model iteratively continues throughout process modeling effort and it involves different verification and validation activities. One such activity is the review of the process by experts and performers.

Before going into the details of process reviews and how they are supported by process modeling tools, let us first take a look at the concepts of process, process model and process documentation, with the help of Figure 1.1. A process is the way people perform their work to achieve certain objectives, e.g., the process of fixing a software defect includes activities, which a software engineer performs to find and fix the issue. As shown in Figure 1.1 the concept of process is normally used as an abstraction of all these complex performance details.
A process model aims to reduce complexity and abstraction by modeling knowledge about required process details. However, it normally fails to reduce the complexity to a level where the model in its entirety is comprehensible by humans. Therefore, to understand a process model, process modeling tools generate renditions of certain parts and details of the process model. These renditions collectively are called process documentation. Each rendition depicts different aspects of the model in different ways and possibly for a different audience, too. For example, process user manuals aim to describe the process for process performers in an easily understandable form.

The primary use of process documentation is process knowledge dissemination. However, as it is usually more understandable and concrete than the process model, it is also used to review the model. Thus a preferred approach is to review the process documentation and to update the model based on these reviews.

Depending on the intended uses of process documentation, process modeling tools use different representation techniques to present different model details. The two most commonly used representation techniques are text-based representation and graphical representation [5]. Both types have their own strengths and weaknesses. For example, a text-based representation can precisely describe process details. However, this precision usually does not come without the use of large amounts of text, which is likely to introduce ambiguities and inconsistencies. Similarly, graphical representations are often easier to understand because of their abstraction, but not good to describe certain details. Consequently, modeling tools try to combine the strengths and minimize weaknesses of both representation techniques [5][6].

1.1 Motivations for the Thesis

As process documentation typically contains large amounts of text, reviewing it can be facilitated by the use of modern word processing tools, which provide features such as commenting and change tracking. Thus, it would be beneficial to have a word processor-compatible rendition of the process model in order to exploit the reviewing and editing capabilities of modern word processors. Figure 1.2 shows how such rendition (Word processing document) can be used to facilitate software process model textual data editing and reviewing.

![Figure 1.2: Textual data review process](image)
Figure shows the generation of word processing document from a software process model. Copies of this document can then be distributed to process model reviewers. Reviewers can use word processing tools to edit and review model’s textual data. Once reviewers send updated copies back to the process engineer, he/she can merge the updated versions and use the final copy to update the model.

Looking at this need to produce word processing documents, some current modeling tools facilitate their generation. However, some lacks in the structure of these documents hinder their review. One issue is the inappropriateness of relationship representation techniques, which results into the difficulty of comprehending the model. Similarly the information in these documents is not stored in a way to support importing of updated document parts back to the process model. This results into a cumbersome activity of manually updating the model.

Currently, no software process modeling tool has the ability to generate word-processor compatible documents that can easily be re-imported into the model. The main problem, as shown in Figure 1.3, could also be seen to represent an n-dimensional software process model in one-dimensional flat document.

![Figure 1.3: N-dimensional to one-dimensional and vice versa](image)

A software process model is considered n-dimensional because it is used to model knowledge about a software process and knowledge itself is considered n-dimensional. In an n-dimensional model entities are related to other entities, which are related to more entities and so on. Thus each entity can relate to n different entities.

Representing such a model in word processing documents is not straightforward, because word processing documents are sequential in nature, where entities can only appear in certain order. This sequential nature of word processing documents makes them one-dimensional because entities can relate to other entities using their sequential order only.
While the transformation of n-dimensional models into n-1-dimensional ones is a known (and solved) problem in mathematics, the process modeling domain poses additional requirements. For example, the one-dimensional representation of the model must still be understandable and editable by human beings, because otherwise the purpose of the transformation is missed. The solution to the problem also highly depends on the context, especially the constraints set by word processors and process modeling tools.

### 1.2 Thesis contributions

With the motivations above, the Thesis develops a mechanism for the representation of n-dimensional software process models in one-dimensional word processing documents. By doing so the Thesis contributes to software process modeling research in following ways:

1) It identifies the issues involved in representing an n-dimensional software process model in a one-dimensional document.
2) It devises a solution for resolving these representation issues.
3) It validates the solution in a software process modeling tool. This allows creation of word processing documents from software process models and the re-importing of reviewed models. Thus providing a powerful reviewing solution for software process model’s textual data.

### 1.3 Thesis Outline

This section provides a brief description about the structure of the Thesis. After introduction of the Thesis in this chapter, Chapter 2 discusses the research approach used in the Thesis. This is followed by some concepts about relationships in Chapter 3. These concepts are used for discussion throughout the Thesis.

Chapter 4 presents the issues of representing n-dimensional software process models in one-dimensional word processing documents. These issues define the direction of the research for developing the solution, which is presented in chapters 5 and 6.

Chapter 5 discusses some basics and constraints of the solution, followed by Chapter 6, which presents the solution for representing n-dimensional software process models in one-dimensional word processing documents.

After presentation of the solution in chapter 6, Chapter 7 discusses static validation of the solution, which is followed by dynamic validation in Chapter 8. Finally Chapter 9 and 10 presents conclusion and future work respectively.

Thesis also includes two Appendices. Appendix I is used to present visual presentation rules. These rules form the core of the solution and are devised to resolve the issues of representing n-dimensional software process models in one-dimensional word processing documents. Appendix II presents a questionnaire used to qualitatively evaluate the usefulness of approach during dynamic validation.
2. Research approach

The research for this thesis has been conducted at the Process and Measurement (PAM) department of Fraunhofer Institute for Experimental Software Engineering (IESE), Kaiserslautern, Germany. The major steps of the research are discussed in section 2.2. However, before going into the details of research approach, section 2.1 lists the research questions used to guide the research.

2.1 Research Questions

The inability of software process modeling tools to represent n-dimensional software process models in one-dimensional word processing documents has lead to the formulation of following research questions.

1) What are the issues related to representing an n-dimensional software process model in a one-dimensional word processing document?
2) How can these issues be resolved in a solution for the representation problem?
3) How can this solution be used by software process modeling tools to facilitate editing and reviewing of model’s textual data?

2.2 Research steps

The three major phases of the research approach followed in this thesis are analysis phase, development phase and validation phase. Figure 2.1 shows these phases with the steps performed in each phase.

Figure 2.1: Research approach
The analysis phase is further divided into three steps, namely analysis of the problem, survey of already available approaches, analysis of constraints on the solution.

The analysis phase is followed by the development phase, which takes into account the findings of the analysis phase and develops the approach for representing software process models in word processing documents.

The development phase is followed by validation phase that is divided into two steps, namely static validation and dynamic validation. The static validation was conducted during the development of the solution through reviews by researchers in the area of process modeling and process reviews, while dynamic validation is performed by implementing the solution in a process modeling tool.

2.2.1 Analysis of the problem

In this step the problem of representing software process models in word processing documents is analyzed. The structure of n-dimensional models is studied in detail and possible issues for representing them in documents are highlighted. These issues are then discussed with researchers in the area of software process modeling at Fraunhofer IESE.

The analysis resulted into a set of seven issues that can hinder the representation of software process models in word processing documents. These issues formed the basis of the research ahead. Chapter 4 discusses these issues in detail.

2.2.2 Survey of already available approaches

The possibility of the existence of solutions for representing n-dimensional models into one-dimensional frameworks resulted into search and analysis of existing techniques. Due to their close proximity the search is conducted only in the areas of semantic web, knowledge management and hypermedia. These areas are selected because they try to represent different types of knowledge in documents and knowledge itself is considered n-dimensional.

The findings of the survey are very positive and resulted into many approaches that can be used to represent knowledge models in documents, especially web-documents. Although these approaches cannot be used in current scenario, due to the constraints of the word processing tools and documents, they provided a basic structure that is used to formulate the solution. This is further discussed in Chapter 5.

2.2.3 Analysis of constraints on the solution

The survey of available approaches provided a basic skeleton that is used by almost all the approaches that try to represent knowledge in documents. However, the use of that skeleton constrained the solution for representing software process models in
word processing documents. These constraints are analyzed during this step of the research. Chapter 5 discusses these constraints in detail.

2.2.4 Development of the solution

The approach for representing n-dimensional software process models in word processing documents is constructed during this step of the research. Attention is given to both human understandability and computer process-ability of the document-contained models. Human understandability is achieved by following a set of nine visual presentation rules. These rules are developed to cater the issues found during “analysis of the problem” step of the research and they are presented in Appendix I.

Computer process-ability is built by annotating document’s textual data with semantically built xml tags. How semantics are built into xml along with other details of the solution is presented in Chapter 6.

2.2.5 Static validation

During the development of the approach it was qualitatively evaluated twice, through reviews by software process modeling experts at Fraunhofer IESE. The reason of using qualitative evaluation is that a quantitative evaluation could only be done through an extensive study, which is easily a thesis itself.

The reviews were aimed to check the comprehensiveness and effectiveness of the solution. Each review resulted into some proposed improvements in the solution, that were carefully analyzed and the solution was updated accordingly. The improvements that resulted from these reviews are discussed in Chapter 7.

2.2.6 Dynamic validation

Once the approach was finalized after two review-update cycles, the next step was to validate the approach dynamically. This was done by implementing the solution in an existing process modeling tool. The Spearmint (Software Process Elicitation, Analysis, Review, and Measurement in and INTOegrated Modeling Environment) software [6] is used for this purpose. Spearmint is a process modeling toolset developed by Fraunhofer IESE.

As with other process modeling tools, Spearmint also lacked the capability of creating word processing documents and to re-import such documents. In order to fill this gap, the developed approach was incorporated into Spearmint.

The implementation enables users to generate word processing documents from models, which are developed in Spearmint. These documents then can be reviewed and edited using MSWord and eventually be re-imported into Spearmint.

The incorporation of the developed approach into Spearmint helped validating its practicability to generate document-contained models, while the computer process-
ability of these models was validated by implementing their re-importing, back to Spearmint.

This was followed by the validation of the usefulness of the approach to facilitate editing and reviewing model’s textual data. For this reason the implemented solution was provided to researchers at Fraunhofer IESE and their feedback was obtained. Dynamic validation is further discussed in Chapter 8.
3. Relationships

A process modeling tool uses a defined set of entity types, relationship types and rules to create a process model [7]. These entities, relationships and rules are collectively called a meta-model. A meta-model is defined only once to generate multiple process model instances [8].

The concept of relationship is used differently both at meta-model level and at model instance level. This is because at meta-model level, a relationship is used to define the type of relationships, while at model instance level each relationship is an instance of a particular relationship type. A single relationship type at meta-model level can result into multiple relationship instances at model instance level.

In addition to this difference in the concept of relationship at different modeling levels, many ambiguities are reported in other relationship related concepts such as cardinality, participation, etc [9][10]. Use of different terminology for same relationship concepts or same terminology for different relationship concepts is not very rare in different conceptual modeling techniques.

In order to avoid such ambiguities in the discussion of relationship representation issues and later to present the approach for the resolution of these issues, following sections define the meaning of some relationship related terms.

3.1 Relationship type and instance

A relationship type represents real-world association among one or more entity types [11]. Some important characteristics of a relationship type are its degree, cardinality, and participation. Figure 3.1 shows a relationship type of “participates in” between entity types “Role” and “Activity”, at meta-model level.

![Figure 3.1: Relationship type and instance](image)

A relationship instance is a single, uniquely identifiable occurrence of a relationship type [11]. Each relationship instance represents an association of entity instances,
one from each participating entity type (Generalized from a binary-relationship instance definition in [12]). Three relationship instances of “participates in” type are presented in Figure 3.1 at model instance level. Each instance connects two entity instances of participating entity types.

3.2 Degree of a relationship

The degree of a relationship type is the number of entity types associated through particular relationship type [11]. Degree of a relationships type is also maintained by every instance of that type, at model instance level. Taking the example of “participates in” relationship type in Figure 3.1 its degree is 2, because it associates two entity types, “Role” and “Activity”. Consequently, each instance of “participates in” relationship also associates only two entity instances, each from a particular entity type.

Degree 2 relationships are also called binary relationships, with degree 3 as trinary. Binary relationships are by far the most common form of relationships [11]. Thus we only will discuss binary relationships and not other higher degree relationships, which are not common in process modeling domain too.

Here it is important to mention that in the subsequent discussion the relationship of an entity type with itself would also be considered as a binary relationship. This is because in such relationships, although the type of the entity remains the same on both sides of the relationship, its role changes on each side.

3.3 Cardinality constraint

Cardinality of a relationship defines the maximum number of instances of an entity type that may relate to a single instance of an associated entity type through a particular relationship type [13]. The basic constructs for cardinality in a binary relationship type are one-to-one, one-to-many, and many-to-many. These are represented by associating signs (‘1’ for one and ‘N’, ‘M’ or ‘*’ for many) with each end of a relationship type, as shown in Figure 3.2. The Figure shows each cardinality construct both at meta-model level and at model instance level. These constructs are explained below:
1. In one-to-one cardinality a single entity instance of a particular entity type can be associated to only one instance of the associated entity type and vice versa.

2. In one-to-many cardinality a single entity instance of the entity type, which is on 1 side of the relationship, can be associated to many instances of the entity type, which is on the N side of the relationship. However, the reverse is not true and each instance of the entity type, which is on the N side of the relationship, can only be associated to only one instance of the entity type, which is on the other side of the relationship.

3. In many-to-many cardinality a single entity instance of a particular entity type can be associated to any number of instances of the other entity type and vice versa.

As shown in Figure 3.2 one binary relationship instance is used to associate only two entity instances, one from each participating entity type. This is different from the concept of relationship at meta-model level, where a relationship type along with cardinality defines a whole set of relationship instances. It also shows that cardinality is a constraint to restrict instances of relationship and associated entities at model instance level.

### 3.4 Participation constraint

Participation constraint specifies whether an entity instance can exist without participating in a particular relationship or not [10]. The participation of an entity in a relationship is defined either as mandatory or optional. Mandatory participation exists when no instance of an entity can exist without participating in a particular relationship. Optional participation on the other hand exists when the instances of an entity can exist without participating in a particular relationship.

Participation is represented at meta-model level by associating number, ‘1’ for mandatory or ‘0’ for optional, with each end of the relationship type. Note that this is
in addition to the sign for the cardinality of the relationship that is also associated with each end of the relationship type. Figure 3.3 shows a convention of writing both participation and cardinality at the end of a relationship type. Using this convention participation is represented on the left followed by two dots, followed by cardinality. This convention will be followed throughout this thesis.

![Figure 3.3: Participation and cardinality](image)

Figure 3.3 also represents the concept of participation and cardinality using the “is managed by” relationship between Department and Employee. The relationship tells that each Department is managed by a single Employee and no Employee can manage more than one Department. This is represented by one-to-one cardinality of the relationship. The mandatory participation of Department, represented by an associated ‘1’ on the Department side of the relationship, tells that no Department can exist without it being managed by an Employee. On the other hand there could be many employees who are not managing any Department, making the Employee participation optional. This is also represented by associating ‘0’ on the Employee side of the relationship.

### 3.5 Categories of relationships

Despite the single dimension of word processing documents, they are capable of representing some types of relationships using their hierarchical structure. This can be achieved in many ways, e.g. by relating entities through the use of heading/sub-heading structure. However, for a relationship to be represented in this way it must also be of hierarchical nature. Keeping this thing in view two categories of relationships are defined for the discussion ahead. These categories are hierarchical relationships and non-hierarchical relationships. Below each of these categories is discussed in detail.

#### 3.5.1 Hierarchical relationships

Hierarchical relationships or parent-child relationships are those relationships in which one entity type acts as a parent or container of an associated entity type. In such relationships if the parent is considered at the \(n^{th}\) level of hierarchy then its
children are at the $n+1^{\text{st}}$ level of hierarchy. This is equivalent to the hierarchical structure of word processing documents, in which one section of a document, at the $n^{\text{th}}$ level of hierarchy, can contain subsections at $n+1^{\text{st}}$ level. These subsections can further contain subsections at $n+2^{\text{nd}}$ level and so on. This structural equivalence makes it possible to represent hierarchical relationships in word processing documents.

Hierarchical relationships consist of three types, namely generalization, aggregation, and property relationships. A brief detail of each of these types is provided below with a purpose to highlight their hierarchical nature.

A **generalization** is a relationship between a general entity type and a specialized form of that entity type [14]. Generalization is also called *is-a-kind-of* relationship. Figure 3.4 shows two generalization relationships one between Entity and Activity and other between Entity and Sub-activity. It shows that both Activity and Sub-activity are kinds of Entity and both share its property of Name. The direction of the arrow is from the specialized entity type to the general entity type.

![Figure 3.4: Generalization, aggregation and property relationships](image)

Generalization relationships might or might not be represented in a model instance. However, they are important to arrange entity types at meta-model level. Also they provide a useful way to structure the word processing documents. This is further discussed in chapter 6.

An **aggregation** is a relationship in which one entity type contains other entity type. Aggregation is also called a *whole-part* relationship, with one entity being a part of another. Figure 3.4 shows an aggregation relationship between Activity and Sub-activity. The direction of the relationship is from the contained entity type to the container entity type, with a special symbol to represent aggregation at the container side.

**Property** relationships are a special form of aggregation. This is because each property can be considered a part of an entity type. However, keeping them as a separate type of relationship is due to the possibility of representing them in different ways in word processing documents. Figure 3.4 shows a property relationship between Entity and its Name. For distinguishing property relationships from other relationship types, a line with an encircled ‘p’ on the entity side of the relationship is used.
Although, the hierarchical nature of above mentioned relationship types make them suitable for presentation in word processing documents, they still pose some issues, which are discussed in chapter 4.

3.5.2 Non-hierarchical relationships

Non-hierarchical relationships are those relationships in which entity types are either at the same level of hierarchy (peers) or they are not related hierarchically. These relationships can also be seen as associations. This is because in the definition of a relationship type, each relationship falls into one basic category of relationships, called associations. Hierarchical relationships are separated from this category, because they pose different issues than associations as a whole. However, associations or non-hierarchical relationships still contain all other relationship types that do not fall into hierarchical relationship category. What types are there in non-hierarchical relationship category is not important, because all of them pose same representation issues.

Representing non-hierarchical relationships in word processing documents is not straightforward. This is because the simple hierarchical document structure, which was found useful in case of hierarchical relationships, does not work for non-hierarchical relationships. Figure 3.5 shows a non-hierarchical relationship of “pairs with” between the entity type of Developer and S/W Engineer in a pair programming scenario [15].

![Figure 3.5: Non-hierarchical relationship](image)

Considering the “pairs with” relationship, it does not look logical to keep both entities related through document’s hierarchical structure. We have to keep both of them either at the same level of hierarchy or in totally different branches in document hierarchy, depending on the overall structure of model. In both cases document’s hierarchical structure does not help us in relating these entities. This marks the need of devising other ways to represent non-hierarchical relationships in word processing document, that can also resolve all the issues discussed in chapter 4.
4. Relationship representation issues

Word processors are widely used as a tool to create textual documents. They provide numerous functionalities for editing and reviewing, including text formatting, spell checking, change tracking, and commenting, to name a few. They also provide styles and templates to structure textual data for better presentation and comprehension.

All these facilities are useful to review software process model’s textual data, using word processing tools. However, understating the meaning of model associated descriptions requires more than their text alone. The context of descriptions is needed too. This context could be made clear by representing relationships between entities, whose textual descriptions are presented in the document.

Being the tools to present plain textual constructs, word processors are limited in their capability to present most of the relationship types. This limitation poses many issues of representing n-dimensional software process models in one-dimensional word processing documents. Below each section discusses one such relationship issue. The issues are related to representing model instances in word processing documents and not about representing meta-models. Therefore, whenever the term relationship is used it means relationship instance if not stated otherwise.

4.1 Representing one-to-one Relationships

One-to-one relationships are the simplest form of relationships that exist both in hierarchical and non-hierarchical categories. Examples of one-to-one relationships of hierarchical category include relationships of entities with their properties, like name, description, etc.

As one-to-one relationships of hierarchical category can easily be represented using document’s hierarchical structure, they pose no issue of representation. Figure 4.1 shows one-to-one hierarchical relationships of Developer and Software Engineer with Description. These relationships can be represented by keeping description as a sub-heading of Developer and Software Engineer entities. This is not only an easy way to represent such relationships, but it is also very understandable for the reviewer.

![Figure 4.1: One-to-one relationships](image)

Figure 4.1: One-to-one relationships
As other non-hierarchical relationships, one-to-one non-hierarchical relationships can exist between entities at the same level of hierarchy (peers) or in different branches of the hierarchy. These relationships cannot be represented using simple and easily understandable hierarchical structures. The “pairs with” relationship, discussed in section 3.5.2 and presented in Figure 4.1, is an example of a one-to-one non-hierarchical relationship. In “pairs with” relationship both Developer and Software Engineer pairs with each other to develop a piece of software. The “pairs with” relationship cannot be logically represented using the hierarchical structure of the document. This is because both Developer and Software Engineer are logically at the same level of hierarchy (peers), if only the existence of “pairs with” relationship type is considered. The problem complicates even further if the existence of other relationship types is also considered, which can put Developer and Software Engineer in totally different branches of the hierarchy. Thus resulting into their placement in totally different parts of the document.

### 4.2 Representing one-to-many Relationships

One-to-many relationships are more complex than one-to-one relationships and they also exist both in hierarchical and non-hierarchical categories. A simple example of one-to-many relationship of hierarchical category is the containership scenario, in which an entity contains multiple instances of a particular entity type. Figure 4.2 shows this scenario, in which an Activity contains Sub-activities.

![Figure 4.2: One-to-many hierarchical relationship](image)

One-to-many relationships of hierarchical category can be represented by arranging children entities under parent entity, with the use of bullets, sub-headings etc. However, the scenario complicates with the existence of too many children. An example is, if the contains relationship, shown in Figure 4.2, allows infinite number of sub-activities to exist in the parent activity. But that is not very common in real world.

One-to-many relationships of non-hierarchical category are more difficult to represent than one-to-one relationships of non-hierarchical category. This is because an entity is involved in ‘n’ more relationships, each of which relates it to other entities in distinct parts of the document.

### 4.3 Representing many-to-many relationships

Many-to-many relationships of both hierarchical and non-hierarchical categories pose representation issues. Hierarchical relationships of many-to-many category are
those relationships in which an entity can have multiple parent entities, while each parent can have multiple children. Figure 4.3 shows this scenario, where parent entity P1 has entities C1, C2 and C3 as children, while parent entity P2 has entities C3 and C4 as children. Entity C3 is the only child with both P1 and P2 as parents.

![Figure 4.3: Many-to-many hierarchical relationships](image)

Such hierarchical relationships can be represented using constructs of heading/subheading, if children entities, such as C3, are repeated under each parent entity. However, this introduces problems of redundancy and inconsistency, which makes this solution infeasible. These problems are further discussed in section 4.6.

Non-hierarchical relationships of many-to-many cardinality are more difficult to represent than one-to-many relationships of non-hierarchical category. Figure 4.4 shows uses type of many-to-many non-hierarchical relationship between activities (Design Architecture, Detailed System Design, Test-case Creation) and artifacts (Requirements Document, Architecture Design Document). Figure shows that each activity can use many artifacts, which could also be the case if uses relationship is a one-to-many relationship. However, the permission that each artifact can be used by many activities makes the representation of many-to-many uses relationships more complex than one-to-many uses relationships.

![Figure 4.4: Many-to-many non-hierarchical relationships](image)

### 4.4 Representing multiple types of relationships

So far the discussion of relationship representation issues is limited only to single relationship type. The cardinality of this relationship can change but no entity is involved in more than one different type of relationships. However, in practice each entity of a process model takes part in more than one type of relationships. Figure 4.5 shows the involvement of Requirements Specification in two different types of relationships namely Create and Uses. Requirements Engineering activity creates
Requirements Specification document, which is then used to design the architecture of the system and to generate test-cases. Without knowing all these relationships of Requirements Specification, it is not possible for the reviewer to understand its purpose.

![Diagram of relationships](image)

Figure 4.5: Multiple types of relationships

Representing all the different types of relationships, in which an entity participates, is one serious issue of representing models in documents. This issue continues to complicate as the number of relationship types increase. This also makes it clear that the issues of representing each category of relationships, i.e. hierarchical and non-hierarchical, cannot be considered as isolated issues. This is because an entity might be involved in more than one, different categories of relationships at the same time. For example, as discussed in section 3.5.1, a simple solution to represent hierarchical relationships is to use a hierarchical document structure. However, if the child entities in a hierarchical relationship are also involved in non-hierarchical relationships with other entities, this solution might not work. Thus a solution to represent one category of relationships must not restrict entities to participate in other category relationships.

### 4.5 Representing Conditions

Models to show flow of data and/or control are very common in software process modeling domain. Most of the issues for representing control/data flow relationships are similar to other issues discussed in this chapter. However, they pose an additional challenge too. Figure 4.6 demonstrates the use of many-to-many relationship to show the flow of control between activities of a software process. The arrow from one activity to another activity shows that the end of preceding activity will start the following activity. The flow is easily understandable from the arrows in the figure. However, if we want to associate conditional information with control flow, the arrows alone would not work.
Conditional information is normally used to put extra restrictions on the flow. For example, we can see that Test-case Creation can start at the end of its preceding activities. Now if we want to put a condition that Test-case Creation can only start after the end of both its preceding activities and not after anyone of them, we cannot do it with simple arrows as used above. This results into the use of complex graphical constructs for representing conditional flows. How these constructs can be represented in a textual document is another representation issue, which needs to be addressed.

4.6 Redundancy

As discussed in case of many-to-many relationships of hierarchical category, one way of representing them is to repeat a child entity under each of its parent entities. Using this mechanism the relationships in Figure 4.3 can be represented in a document as:

\[
P1: \\
C1 \\
C2 \\
C3 \\
P2: \\
C3 \\
C4
\]

As C3 is a child of both P1 and P2, it is repeated under both of them. This solution does not seem practical due to two reasons. Firstly, it introduces redundancy. Redundancy subsequently creates the problem of updating an entity’s associated text at multiple places, which opens the possibility of inconsistencies in the document.

Secondly, it does not facilitate reviewer in getting the context of C3 at one place. This is because each relationship, in which C3 participates, is represented in some distinct part of the document. A reviewer must find all these relationships to understand C3’s context.

Despite these flaws of repeating an entities textual data at multiple places in the document, it is sometimes very useful and simple to do this. An example is the relationship of an entity with its type. As shown in Figure 4.7 both Design Architecture and Detailed System Design have the type of Activity. In a graphical representation all the entities of type activity are linked to the single entity Activity.
However, in a word processing document it is more understandable if the type of each entity appears with its description, repeating each type multiple times.

![Diagram of entities with their type]

**Figure 4.7: Relationship of entities with their type**

When the use of redundancy achieves simplicity and when it creates ambiguities, must carefully be analyzed while purposing a solution for representing software process models in word processing documents.

### 4.7 Computer process-ability

A software process model represented in a word processing document should not only be understandable by humans but it also needs to be computer process-able. The major purpose is to automatically re-import model textual data back to the process modeling tool. This highlights the need to represent models in document’s storage format as well. In this way the process modeling tool can recognize and re-import model constructs by parsing word processing documents.

Being able to visually represent models in word processing documents does not ensure their computer process-ability. This is because meanings conveyed by document’s visual layout, could be understandable by humans but for word processors it is merely some structured text. The problem further complicates because we cannot change available word processor storage formats or the way these formats are used to store documents.

How software process models can be represented in word processing documents, so that they can be re-imported by process modeling tools, is an important representation issue. Also, as the solution to this issue can impact the choice of solutions for other representation issues, it is necessary to resolve this issue before considering others.
5. Basics and constraints of the solution

A process model is a semantic construct, in which entities are linked through relationships to convey knowledge about process. Different model representation techniques can use different ways to represent model’s entities and relationships, but the knowledge it conveys remain computer process-able as long as the model’s semantic meanings remain intact. The same is true for keeping the model human understandable. However, model’s human understanding also depends on the clearness, with which different concepts are visually presented, and human’s own comprehension capabilities. Keeping these things in view, the problem of representing models in word processing documents can be seen as a problem of building computer and human understandable model semantics in word processing documents.

5.1 Three-step-methodology

The idea of building semantics in textual documents is not new. This can be seen from the existence of number of techniques [16][17][18][19], developed during last couple of decades. The areas under which this research has been conducted include semantic web, knowledge representation, hypermedia, etc.

Despite the versatility of the research areas in which these techniques are developed and the possible difference in their targeted audience (humans or computers) these techniques exhibit some commonalities. Many of these techniques share three basic steps for building semantics in textual documents. These steps are:

1. Modeling real world concepts
2. Building rules from concepts
3. Generating semantic documents based on rules

Before representing real world concepts in documents they are first modeled through different modeling techniques such as object-oriented design, ontology design, etc. Once the concepts about particular world domain are modeled, the next step is to represent these concepts in the form of computer and/or human understandable rules. The computer understandable rules can be built using computer process-able languages such as Document Type Definition (DTD), Extensible Markup Language Schema (XML Schema), Resource Description Framework Schema (RDF Schema), etc. While human understandable rules are normally the rules on how to visually represent entities and their relationships e.g. in case of hypertext a relationship between two entities can be represented with a hyperlink.

The rules developed, both for computer and/or human understandability, are used to generate the document. Once developed, the rules can be reused to generate multiple document copies, each having different data but exhibiting common semantics.
5.1.1 Examples of three-step-methodology

M Erdmann et al in [16] presented an approach for building semantics in XML documents. The authors argue about lack of capability of XML in building true semantic documents and provided an approach to deal with this limitation. The approach models real world concepts using ontology. The ontology is then used to derive XML Document Type Definition (DTD), which is used to instantiate truly semantic XML documents. Designing the ontology, deriving DTD, and instantiating documents based on DTD, exhibits that the approach is based on three-step-methodology of building computer understandable semantics in XML documents.

Nanard J. et al, in [17] discussed an approach to resolve the problem of user disorientation in hypertext. They attributed this problem to the lack of conceptual model in hypertext and proposed an object oriented model for better human navigation. The object oriented model is used to build conceptual view of a specific domain of knowledge. This model is then mapped onto hypertext, with model relationships represented by hyperlinks and model concepts by chunks of information on the web.

The three explicit parts of the approach are 1) designing object oriented model, 2) relating the concepts in the model to visible constructs like hyperlinks, and 3) generating hypertext documents using visible constructs. These three parts puts this approach in the category of three-step-methodologies for building human understandable semantics in hypertext documents. In addition the authors also discussed usefulness of the approach for computer process-ability.

5.2 Advantages of using three-step-methodology

Being the core of too many semantic building techniques, three-step-methodology lends itself as a base, on which the approach for building model semantics in word processing documents could be established. Some of the advantages it offers in current scenario are:

1. **No need to reinvent:** The techniques of building semantics in textual documents are developed in different areas of research during a span of more than twenty years. Yet their fundamentals are based on three-step-methodology. This exhibits the maturity of the methodology. Therefore, rather than defining an approach from scratch it is better to use the already available skeleton of three-step-methodology and customizing it with particular requirements.

2. **Use of already developed standards:** Another advantage is the possibility of using already developed standards that different approaches use at each step. For example for defining semantic rules, standards such as RDF Schema can be used. Similarly, mechanisms such as RDF or XML can be used to generate documents. However, the choice of using a standard depends on the capabilities of word processors.
3. **Reuse of already available solutions:** There is a possibility that some or all of the issues of representing models in word processing documents, might already be resolved in one available approach or another. The possibility of reusing these solutions greatly enhances the advantages of using three-step-methodology.

### 5.3 Requirements for using three-step-methodology

The advantages of using three-step-methodology can greatly facilitate the research for representing process models in word processing documents. However, at the same time, the possibility of reusing any existing technique, which is based on three-step-methodology, is rare. The reason is the limitations posed by word processing tools.

In order to base the approach, for representing process models in word processing documents, on three-step-methodology, the word processing tools must fulfill a specific set of requirements. These requirements are:

1. **Open standard format:** The format in which the documents are saved must be open standard. Only then it can be parsed by tools other than word processors.

2. **Use of formats that support semantic building:** Storage format of a document is very important to build computer understandable semantics. Some formats like XML support this capability.

3. **Support of building semantics for multiple domains:** Word processors use concepts such as headings, bullets, paragraphs, font sizes, etc to distinguish between different parts of the text. All these concepts are from the domain of formatted text and they are used to associate visual meanings with text. This indicates that word processors already generate semantic documents. However, representing concepts of process modeling domain puts an extra requirement on word processors. This is the requirement of representing concepts from more than one domain. One domain is of course the formatted text domain and the other is process model domain. A word processor must represent concepts from both domains at the same time.

4. **Keeping model semantics intact:** The basic reason of building computer process-able semantics in word processing documents is to provide facility of model re-importing. However, this can only be done if word processors do not change model semantics while saving an updated version of the document. This can be achieved by making word processors aware of model semantics, by disallowing model semantic updates, etc. Whether word processing tools support any such mechanism or not, is an important question.
5.4 Selection and evaluation of a word processor

Since the advent of WYSIWYG (What you see is what you get) software-based word processors in early 1980s [20], many word processors become available in market. The major aim of these word processing tools is to provide editing, composition, formatting, etc of printable material [20]. However, they differ a lot in the way they offer these features. This can be seen from the available comparisons of different word processing tools [21][22][23][24].

Due to the lack of any common standard for available word processing tools, it does not look feasible that more than one word processing tools can fulfill the requirements of using three-step-methodology in same way, if they are fulfilling them at all. This infeasibility, limits development of an approach that fits for variety of word processing tools. Thus we have to choose one tool and search its capabilities for fulfilling above requirements of using three-step-methodology.

Being the most widely used word processor [20], Microsoft Word (MS Word) is one clear candidate. A major contender is OpenOffice’s Writer – Writer is an open-source word processor and is rapidly gaining popularity [20]. However, the purpose of developing the approach is not only to provide a solution for representing process models in documents, but to make it usable for a bigger audience too. This leads to the selection of MS Word as a target word processor for the development of the approach.

Before defining the approach for representing process models in MS Word documents, it is necessary to conduct a research on whether MS Word fulfills above mentioned requirements or not, and how it fulfills them. This also defines a set of constraints on the approach for representing process models in MS Word documents.

1. **Support for XML:** Since 2003, Microsoft Word started supporting documents with XML format [25]. These documents can be read, transformed, and manipulated not only by MS Word 2003, but by other XML tools as well [26]. In addition XML can also be used as a tool for building semantics in documents, if the document structure is based on a conceptual model [16]. These capabilities fulfill first two requirements for using three-step-methodology, namely the requirements of open standard format and support for semantic building. However, at the same time it restricts to use only XML for representing models in MS Word 2003 documents.

2. **Support for custom schema:** A Microsoft Word 2003 XML document is based on WordML (XML format for MS documents) schema, which defines its structure [26]. In addition to this schema, MS Word 2003 also supports use of user-defined schemas [26]. This custom schema facility can be used to build concepts of multiple world domains in MS Word 2003 documents. This can be done by associating custom schemas with MS Word 2003 document and applying schema tags to the data in the document [26]. Another advantage is that MS Word 2003 keeps these tags intact, while performing document text editing. At the same time it provides special editing modes to change those semantics and to prompt user about the problems related to specific custom schema (For more details on the use of custom schemas in
MS Word 2003 see [25]). These facilities fulfill third and fourth requirement for using three-step-methodology, namely the requirements of supporting semantics for multiple domains and keeping the model semantics intact. However, at the same time they restrict to use only XML-Schema for building computer process-able rules in MS Word 2003 documents.

### 5.5 Constraints of the approach for representing models in documents

The findings in this chapter define following set of constraints on the approach for representing software process models in word processing documents. These constraints along with relationship representation issues, discussed in chapter 4, form the basis of the approach, discussed in next chapter.

1. The approach will be based on three-step-methodology for building semantics in documents.

2. Due to lack of common standard for available word processing tools, the approach will aim to represent software process models in MS Word only.

3. Since the requirements posed by three-step-approach are only fulfilled by MS Word 2003 and later versions, the approach would only work for these versions.

4. Microsoft’s WordML format will be used to generate MS Word documents.

5. XML-Schema will be used to build computer process-able rules in the second phase of the three-step-methodology.

6. MS Word 2003’s custom schema facility will be used to build process model semantics in generated WordML documents.
6. Representing SPMs in MSWord documents

The approach for representing Software Process Models (SPMs) in WordML documents is based on the constraints defined in section 5.5. Figure 6.1 shows the contents of each step of the approach. Boxes in each step represent the action unit/s of that step, while circles represent either the rules (light shade circles) or the data (dark shade circles) that is used by the action units to generate outputs. The outputs are represented by outward arrows from action unit.

![Figure 6.1: Three steps of the approach](image)

Step 1 creates a meta-model of the process domain, which is created based on a set of meta-modeling rules. This meta-model is used as an input to generate XML-Schema in Step 2. The generation of XML-Schema is also based on Visual Presentation rules (VP rules) and schema generation rules. Creation of visual presentation rules is also a part of Step 2. Once visual presentation rules are created and they are used to generate XML-Schema in Step 2, they are also used along with generated XML-Schema for annotating WordML documents in Step 3. The annotation of WordML document is also based on WordML Schema, and process model data.

Although the approach uses four different sets of rules, the most important of them are the meta-modeling rules and visual presentation rules. This is because both these sets of rules try to resolve relationship representation issues. The visual presentation rules are presented in Appendix I, while meta-modeling rules are presented in section 6.1, along with create meta-model step. This is followed by build rules step in section 6.2, and annotate document step in section 6.3.
6.1 Create meta-model

The major aim of this step is to create a meta-model of the software process domain. A meta-model defines entity types, relationship types and the rules to use these constructs for creating process model instances.

This section provides a set of rules for creating a meta-model. These rules must be followed for the creation of any meta-model that needs to be used in this approach. The rules are presented in the form of meta-meta-model. A meta-meta-model is used to define how different meta-models can be constructed, just as a meta-model is used to define how a model instance can be created. A meta-meta-model is another model abstraction level in addition to meta-model and model instance levels. It lies on top of meta-model level and a single meta-meta-model is used to generate multiple meta-models, each of which is then used to generate multiple model instances.

Figure 6.2 shows a meta-meta-model, which is based on the basic relationship concepts, discussed in chapter 3. Looking at the rules, defined in Figure 6.2, a meta-model can consist of entity types, properties, and relationships. The types of relationships include generalizations, associations, aggregations and property. Generalizations, associations, and aggregations support all forms of cardinality, while properties can only support one-to-one and one-to-many forms. In addition, participation of an entity in any particular relationship type is not mandatory.

![Figure 6.2: A generic meta-meta-model](image)

Due to its generic nature, the meta-meta-model presented in Figure 6.2 can be used to define a vast majority of meta-models. However, it does not define any rules to avoid relationship representation issues. If any such rule can be defined, it can help avoiding the existence of relationship representation issues in meta-models, which consequently result into the resolution of these issues in model instances. Keeping this thing in view, section 6.1.1 defines some constraints, which disallow the use of many-to-many hierarchical relationships in meta-models.
6.1.1 Constraints on many-to-many hierarchical relationships

The physical structure of WordML or generally speaking XML documents is a hierarchically ordered set of entities [27]. Although this structure is very useful to represent one-to-one and one-to-many hierarchical relationships, many-to-many hierarchical relationships still pose some issues.

The resolution of many-to-many hierarchical relationship representation issue is not straightforward. However, the exclusion of many-to-many hierarchical relationships from a meta-model is possible. This can be achieved by putting some constraints on the hierarchical relationship types at meta-meta-model level. These constraints are defined as Constraint 1 and Constraint 2 below.

**Constraint 1:** Many-to-Many form of generalization relationships are not allowed in the meta-model. This is achieved by not allowing the specialization of an entity from more than one entity types. Figure 6.3 presents this constraint, by making generalization a one-to-many form of relationship type.

**Constraint 2:** Many-to-many form of aggregation relationships would not be considered as hierarchical relationships. Rather they will be dealt as many-to-many non-hierarchical relationships or associations. This is also constrained by meta-meta-model shown in Figure 6.3.

![Figure 6.3: Constrained meta-meta-model](image)

After applying both the constraints, the representation of all hierarchical relationships in WordML documents become possible and that too without losing too much generality. The only loss in generality is the inability to specialize from multiple entity types, or in other words the inability of multiple-inheritance. However, multiple-inheritance has its own disadvantages, some of which can cause semantic problems [28]. In addition, the relations modeled through multiple-inheritance can also be modeled using single-inheritance, without losing their semantics. Some
techniques for transforming multiple-inheritance to single inheritance are discussed in [29] and [30].

Similarly, the constraint that many-to-many aggregation relationships can not be modeled as aggregations does not affect the generality of the meta-meta-model too. This is because same relationships are allowed to be represented as associations. Therefore, even though the many-to-many aggregations are not allowed, their semantics can still be modeled. In addition, the containership of a child entity in multiple parent entities is not a common phenomenon.

The rules presented in Figure 6.3, are the final set of rules for creating a meta-model. These rules must be followed by every meta-model that needs to be used in current approach.

6.2 Build rules

The major aim of this step is to generate computer process-able and human understandable rules. The computer process-able rules are in the form of XML-Schema, while human understandable rules are in the form of visual presentation rules. Figure 6.4, presents the internals of build rules step.

Figure 6.4: Build rules step

Figure 6.4 shows that the visual presentation rules – or human understandable rules – presented in Figure 6.1 as one set of rules consist of two sub-sets, namely document layout rules and text formatting rules. Document layout rules are used within Step 2
to generate XML-Schema, while text formatting rules are used in Step 3 to format process model textual data.

The creation of visual presentation rules is discussed in section 6.2.1, while the generation of XML-Schema is discussed in section 6.2.2.

6.2.1 Create visual presentation rules

Visual presentation rules are those rules that define how entities and their relationships can be visually represented in a WordML document so that the model’s semantics can be understood by humans. Defining these rules is one major objective of build rules step. In order to achieve this objective the visual presentation rules must not only provide a way to represent entities and relationships, but a major challenge is to make the representation easily comprehensible by humans. In addition most of the relationship representation issues, presented in chapter 4, also relate to visual presentation of relationships. Thus visual presentation rules must also provide ways to resolve these issues.

Keeping all these challenges in view Appendix I provides a set of nine visual presentation rules. These rules are formulated for use with any model, which is based on the meta-model created in Step 1 of the approach. This is achieved by providing presentation rules for all basic relationship types, namely generalizations, associations, aggregations, and property relationships.

Despite the usefulness of rules presented in Appendix I for a vast majority of models, the approach does not restrict us to use these rules as it is. The reason this restriction is not build into the approach is the dependency of human understandability on visual presentation rules. The visual presentation of textual data that is very useful for one set of audience might not make too much sense for another audience.

However, the flexibility of being able to change the rules, presented in Appendix I, does not undermine their importance. This is due to the following reasons:

1. The rules are developed to work with all models, which are based on the meta-model created in Step 1 of the approach.
2. The rules provide solution to almost all the relationship representation issues, presented in chapter 4.

By focusing on these points an attempt is made to keep the need of change in Appendix I rules to minimal.

Once the visual presentation rules are finalized they are used as input both for generating XML-Schema in Step 2 and for annotating WordML document in Step 3. For this purpose they are divided into two parts as shown in Figure 6.4. Document layout part is used within Step 2 and it defines how to structure the document hierarchically, while text formatting part is used in Step 3 and it defines how the rule formats the text. Each visual presentation rule in Appendix I have both these parts.
As the document layout parts are used in Step 2 to generate XML-Schema, they are joined together to form a document layout model in Figure 6.5. Document layout
model represents how basic relationship types, i.e. generalization, association, aggregation and property relationships, can be represented in a hierarchical way using visual presentation rules. It also shows how visual presentation rules use property and aggregation relationships to replace associations, which makes a process model presentable in single-dimensional word processing documents.

6.2.2 Generate Schema

The generate schema action unit is divided into two sub-steps, as presented in Figure 6.6. In the first step the document layout rules are applied on meta-model to generate a document layout model instance. This sub-step is further explained in section 6.2.2.1. The second step transforms the document layout model instance into XML-Schema by applying model to schema conversion. This transformation is performed by applying schema generation rules on document layout model instance. This sub-step is further explained in section 6.2.2.2.

Figure 6.6: Sub-steps in generate schema action unit

6.2.2.1 Apply document layout rules on meta-model

The document layout model, presented in Figure 6.5, is the model of the structure of a word processing document. This structure is defined by visual presentation rules and it provides a way to represent entities and relationships in a single-dimensional word processing document. However, a meta-model, created in Step 1 of the approach, is an n-dimensional model and as long as it does not follow the document layout model pattern, it cannot be represented in a word processing document. This marks the need to convert the meta-model to an instance of document layout model by applying document layout rules on it.

In order to convert a meta-model to document layout model instance the document layout part of visual presentation rules are applied on it in sequence. Starting from rule number 4 and ending at rule number 9. In addition the first three rules, i.e. rule number 1, 2 and 3, are followed whenever an entity, entity section, property section, association section, or sub-entity section is instantiated.

Document layout model instantiation itself is not very difficult to understand, but the fact that each entity type in a process meta-model result into a complete replication of document layout model makes the resulting instance look complex. This can be seen from contents of light gray boxes in Figure 6.8. Figure 6.8 represents an instance of document layout model. This instance is generated based on the meta-model presented in Figure 6.7.
It can also be seen that the complexity of document layout model instance not only depends on the number of entity types, but also on the number of types of particular associations, and types of properties in the meta-model. This is because each new type of these constructs could produce a complete replication of its contained constructs. This is shown with the use of dark gray boxes in Figure 6.8. Normally a meta-model contains many of these constructs and thus it results into a very complex document layout model instance. Representing such a complex model instance graphically might not help understanding the instance in a better way. Thus the graphical representation of document layout model instance can be skipped.

It is important to mention here that by recommending the skipping of graphical representation of document layout model instance, the approach does not allow the skipping of the whole step of applying document layout rules on meta-model. Instead the graphically representation can be skipped and replaced by schema representation. Schema generation is discussed in section 6.2.2.2.
Figure 6.8: Document layout model instance
6.2.2.2 Transform document layout model instance to XML-Schema

The major aim of this sub-step is to transform the document layout model instance into XML-Schema. This transformation is performed by applying schema generation rules on document layout model instance. Schema generation rules define the mapping between document layout model constructs and XML-Schema constructs. Document layout model constructs are entities, relationships, and properties, while XML-Schema constructs are complexType, simpleType, element, attribute, etc.

Different approaches for transforming XML document models to XML-Schema exist in literature [31][32]. These approaches take different types of XML document models as input and generate an XML Schema based on these models. In addition, there are XML-Schema generation tools [33][34] that provide simple graphical constructs to model an XML document and then generate XML-Schema from that model.

Due to the rich collection of available approaches and tools for XML-Schema generation, current approach does not provide any new set of XML-Schema generation rules. Rather it recommends the use of rules presented in “A Semantic Network-Based Design Methodology for XML Documents” [32]. Although the technique uses semantic network-based models as input for generating XML-Schema, the schema generation rules themselves are quite flexible. Therefore, they can be easily used to transform the document layout model instance into XML-Schema.

The most important advantage of using recommended rules for generating XML-Schema is their capability to transform all four basic relationship types, namely generalizations, aggregations, associations and property relationships. In addition, the rules also map the hierarchical structure, represented in document layout model instance, to XML schema with the use of nesting of entities. Once XML-Schema is created, it is used in Step 3 to generate WordML documents.

6.3 Annotate document

MSWord 2003 uses a number of schema namespaces to build a WordML document. These schemas provide sets of tags to define the visual structure of textual data, such as its format. In addition, the custom defined XML-Schema tags are used to structure textual data according to its semantic meanings.

In the current approach, the formatting tags are applied according to the text formatting rules that are part of the visual presentation rules. Similarly, the custom schema tags are applied as defined in XML-Schema, also generated in Step 2 of the approach. The discussion in this section does not go into the use of different tags. However, it explains how custom schema and formatting schema tags can be interleaved to generate a WordML document. This discussion is useful for creating WordML documents from applications other than MSWord 2003. For annotation of textual data using MSWord 2003 facilities, [25] provides an example.
6.3.1 Interleaving of custom schema tags and formatting tags

Although a WordML document consists of many parts, the area where custom schema tags are used together with Word’s schema tags is called the *Body* of the document. The *Body* contains the textual data of the document. Figure 6.9 shows the *Body* of a WordML document when only the custom schema tags are applied. The custom schema tags include EntitySection, Name, and Description. These tags are part of the XML-Schema generated in second step of the approach and that is referred as “ns0” namespace here.

```xml
<w:Body>
  <ns0:EntitySection ns0:type="Activity">
    <ns0:Name>
      Activities
    </ns0:Name>
    <ns0:Description>
      This section contains all the Activities.
    </ns0:Description>
  </ns0:EntitySection>
</w:Body>
```

Figure 6.9: Body of a WordML document without formatting tags

Figure 6.9 shows how model’s textual data can be annotated with custom defined schema tags in WordML documents. In addition, Figure 6.9 also shows the representation of one-to-one property, called *type*, as attribute of EntitySection tag.

Once the textual data is annotated with custom schema tags, the data become computer process-able. However, it does not become human understandable as long as the data is not annotated with WordML’s formatting tags too. These tags need to be applied in a way that makes the visual presentation of textual data according to the visual presentation rules presented in Appendix I. Figure 6.10 shows the annotation of textual data using formatting tags, in addition to custom schema tags.
Figure 6.10: Body of a WordML document with formatting tags

Figure 6.10 presents how textual data is annotated with WordML’s formatting tags before the whole block, consisting of both textual data and formatting tags, is put into custom schema’s start-end tags. As the whole document is structured in this way, the document become both computer process-able and human understandable.
7. Static validation

The static validation was aimed to assure that the approach was in good shape before it could be validated dynamically. Therefore, the improvement suggestions that resulted from static validation were incorporated into the approach before its dynamic validation. It is important to mention that the approach presented in Chapter 6 is the final version of the approach after the incorporation of changes, suggested during static validation.

Static validation was performed twice in the form of reviews of the approach, collecting the comments from two different reviewers, each an expert in the area of software process modeling and reviews at Fraunhofer IESE. A documented version of the approach was provided to the reviewer, who went through the findings of the research and provided his feedback, both orally and in written form too.

The reviews were not conducted on same versions of the approach. Rather the feedback of first review was obtained, analyzed and the approach was updated before conducting the second review on the improved version of the approach. The aim of using an improved version in second review was to allow the reviewer to concentrate on those aspects of the approach that might had been missed in the first review. Otherwise the results of both the reviews could overlap and some problems, which were not visible because they were hiding behind some minor issues, could be missed.

The major suggestions that resulted from both reviews are discussed below along with the rationales on why the suggestions were used or not used to amend the approach. The suggestions are discussed in order of their impact on the approach and not in the order of reviews. However, discussion of each suggestion provides the review number in which it was suggested.

7.1 Restrictions to use Appendix I rules

The second review resulted into some suggestions about visual presentation rules, presented in Appendix I. The most important of them was the suggestion that despite the usefulness of the rules for a vast majority of models and their effectiveness to resolve relationship representation issues, the rules could not be considered final for all scenarios. The major reason provided was the dependency of human understandability on visual presentation rules. Reviewer’s, opinion was that minor changes in the rules are expected to make them usable for different audiences.

This was a good suggestion and in order to make the approach usable in different scenarios the approach was updated. The major changes were to remove the dependencies on the already defined set of visual presentation rules. This was done by defining the “create visual presentation rules” action unit in the approach, as shown in Figure 6.1. This action unit provides a way to amend visual presentation rules or to redefine them before their use.
Prior to the existence of “create visual presentation rules” action unit the approach was dependent on to use of Appendix I rules in all scenarios.

7.2 Model section is needless

A comment from the second review concerned about the Model section as the root section in the document layout model. The Model section was used to group entity sections in the document and consequently it introduced an extra level of hierarchy in the document. The reviewer suggested dropping the Model section and making entity sections as the root sections of the document.

The suggestion was implemented, as it depicts one important aspect of a good word processing document, i.e. to keep the levels of hierarchy low. This was complimented by the fact that dropping the Model section did not lose any information about the model. These arguments lead the way to exclude Model section from visual presentation rules and hence from the document layout model as well.

7.3 Sub-entities section is needless

Another suggestion from the second review was the removal of sub-entities section from the visual presentation rules and to present sub-entities directly under the heading of the entity. This suggestion was also based on the fact that keeping sub-entity section introduces an extra level of hierarchy in the document.

However, at the same time it implicitly suggested the placement of sub-entities at the same level of hierarchy as property and association sections, thus making it difficult to distinguish between sub-entities and other sub-sections. This is especially true if sub-entities are not put at the end of an entities text-block, with property section and association section before them. This can happen because visual presentation rules do not build any restrictions on where to put different sub-sections within an entity’s text-block. Such restrictions can be built, but that is not without the loss of some generality of the approach. Therefore to avoid this loss this suggestion was not incorporated into the approach.
8. Dynamic validation

The dynamic validation of the approach was performed to achieve the following objectives:

1. To validate the practicability of the approach for generating WordML documents.
2. To validate computer process-ability of generated WordML documents.
3. To validate the usefulness of the approach, for facilitating software process model textual data editing and reviewing.

The first dynamic validation objective was achieved by going through the steps of the approach using the scenario of a software process modeling tool, called Spearmint [6] (Software Process Elicitation, Analysis, Review, and Measurement in and INTEGRATED Modeling Environment). Spearmint is a process modeling toolset developed by Fraunhofer IESE.

A meta-model was created in the first step of the approach based on the modeling concepts of Spearmint. This meta-model was used along with the visual presentation rules of Appendix I to generate XML-Schema in the second step of the approach. This was followed by the creation of annotated WordML documents in the third step of the approach. The creation of annotated documents was implemented in Spearmint, which enabled it to generate WordML documents from software process models, designed in the tool. The validation results obtained from all these steps are discussed in section 8.1.

The second objective was achieved by implementing the importing of generated WordML documents back to Spearmint. The results of this validation step are discussed in section 8.2.

The third dynamic validation objective was achieved by obtaining a feedback about the usefulness of the implemented solution from the researchers in the area of software process modeling and reviews at Fraunhofer IESE. One important decision was interview those researchers that were already familiar with Spearmint and used it for modeling and reviewing of software process models.

Two researchers participated in the feedback, which was obtained through the use of semi-structured interviews. Prior to each interview, a presentation session was conducted to present the solution to the interviewee. The presentation of the solution was quite informal and was aimed to resolve ambiguities and answer questions of the interviewee.

After the presentation, the interview was conducted using the questionnaire in Appendix II. The questions in the questionnaire were designed to be open-ended, because their purpose was to initiate discussion on a certain aspect of usefulness and not to measure it on a scale. The feedback obtained from the interviews is presented in section 8.3.
8.1 Validation of practicability of the approach

Applying each step of the approach using a Spearmint scenario helped validating the practicability of the approach in a real world scenario. The findings of this validation step are discussed below.

8.1.1 Limited space for representing multiple levels of sub-entities

One limitation that was observed while generating WordML documents from software process models in Spearmint was the lack of space to represent sub-entities that were on third or greater levels of containership. This was because each level of hierarchy (subheading) was horizontally indented by default to keep it visually distinguishable from other levels. This indentation was found very useful for showing containership relationships of first and second level. However, if second level sub-entities further contain sub-entities, it become impossible to keep them indented, due to limited horizontal space of the document.

This problem was first resolved by putting a constraint that sub-entities, which were at third and higher levels of containership, would not be represented in the document. However, as this limitation was quite restrictive, advice was sought from reviewers that were interviewed about the usefulness of the approach. Their suggestions are discussed in section 8.3.7.

8.1.2 Lack of support for formatted data representation

A finding while implementing the approach in Spearmint was that the process modeling tools not only support the association of plain text with entities, but they also support formatting of that text to present it in a better way. The text appears as formatted text, while it is saved with the use of HTML formatting tags to annotate textual data.

Although these tags are used for formatting the text and they are meaningful for software process modeling tools, their existence in WordML documents does not give a proper format to the tag-contained data. This is because WordML uses different tags for same formatting concepts.

Although the finding is important, it does not pose any threats for the practicability of the approach, instead it makes the implementing more challenging. The implementer of the approach has to develop the conversion mechanism from software process modeling tool’s text-formatting tags to WordML text-formatting tags and vice versa.

8.1.3 Limitation to use textual-data outside of custom-schema tags

A problem was observed while implementing the annotation of WordML documents in Spearmint. This was related to the use of textual constructs that were not defined in custom XML-Schema and their only use was to present model textual data in a
proper way. One such example was the use of commas to separate relationship instances in one cell of a relationship table.

The problem was resolved by defining tags for non-model constructs such as separators and the use of those tags to annotate non-model textual data in WordML document.

8.2 Validation of use of approach for generating and importing WordML documents

This step was used to validate computer process-ability of WordML documents generated from Spearmint during the first dynamic validation step. The document was parsed for Spearmint schema tags and model’s textual data was extracted and imported back to Spearmint model.

No major problems were observed during this validation step. However, some minor problems related to the implementation details were observed. As these problems do not threaten the application of the approach they are not discussed here.

8.3 Validation of usefulness of the approach

The usefulness of the approach to facilitate software process model textual data editing and reviewing was validated by obtaining feedback about the usefulness of the implemented solution. This feedback was obtained in a qualitative way through interviews with researchers in the area of software process modeling and reviews at Fraunhofer IESE.

The contents of the questionnaire that was used to conduct the interviews are presented in Appendix II. The questions in the questionnaire were designed to initiate discussion on certain aspects of usefulness such as the usefulness of representing different entity types, relationship types, etc, in the way they are represented in Spearmint generated documents. As this representation was completely based on Visual presentation rules presented in Appendix I, no example of Spearmint generated document is provided here. Some questions for comparing the usefulness of implemented solution with already available editing and reviewing facilities in Spearmint were also asked.

Below each section discusses the feedback obtained from the interview and thus provides a way to analyze the usefulness of the approach for editing and reviewing software process models textual data. As the questions were asked to qualitatively validate the usefulness of the approach, the results do not give any idea about the exact improvement in usefulness. This deficiency could be resolved by conducting a quantitative validation of the approach. However, quantitative validation could itself be a full thesis and therefore could not be conducted in the time constraints of this thesis.
8.3.1 Representation of properties

The first question was asked about the representation of properties in the document. The general view was that the representation is good but can be improved by minor changes.

Both interviewees commented to improve the format of heading and description properties. Although these suggestions were quite useful to improve the representation of properties in Spearmint they did not put any threat for the usefulness of the approach in general and the visual presentation rules in particular. This was because the visual presentation rules do not restrict the use of any formatting styles for properties.

Another comment was that the visual presentation rule to present one-to-many property relationships in a table may not always result in a good structure. The reason pointed out was the possibility of too much text associated with each or some instances of one-to-many property relationship. This can result into a table that can spread over many pages of the document, which of course will look odd.

The recommendation to resolve this problem was to represent each instance of one-to-many property relationship as one-to-one property relationship. However, the practicality of this recommendation becomes limited, because it can result into ‘n’ property sections, one for each instance of one-to-many property. This also can result in similar problems as with using tables to represent one-to-many property relationships.

As there is no clear solution to this issue, it still remains open for discussion and will be further investigated in future research work.

8.3.2 Representation of hierarchical relationships

The second question was about the representation of hierarchical relationships. The general view was that the containership of text sections with in other text sections gives a clear view of the hierarchical relationships. However, the use of indentation to represent containership was not encouraged. Rather both the interviewees recommended removing the indentation and using only heading numbers and font properties to represent containership as it provides a smooth layout. This was also supported by the argument that MSWord provides an indented view of the document sections in Document Map. Therefore, it is better to use that and keep the document un-indented.

As the use of indentation is not restricted by the visual presentation rules, no changes in the approach were needed. However, the recommendation was very useful for improving the implementation in Spearmint.

8.3.3 Representation of associations and flow-conditions

Both the interviewees liked the use of tables to represent associations. However, both of them recommended the use of hyperlinks on both sides of the directed
associations. The rule VP.R8 asks for the hyperlinks to appear only in the text block of the source entity and not in the target entity. From interviewee’s point of view this results into lack of navigation from destination entity to source entity. The observation is quite right and marks the need to amend rule VP.R8, which will be done in next version of the approach.

The representation of flow-conditions with the use of brackets and binary operators was also appreciated by the researchers.

8.3.4 Navigation through the model

Except from the absence of hyperlinks on both sides of the directed associations the support for navigation of the model was appreciated by the interviewees. They also highlighted the usability of document map facility of MSWord, for navigation.

8.3.5 Model clarity

Question 6 in the questionnaire was asked to check the clarity of the model as a whole. It questioned the need of using graphical representation to understand the model before it can be reviewed using MSWord facilities. Both the interviewees pointed out that even though MSWord’s editing and reviewing functionalities are very helpful to edit models textual data, understanding models from purely textual documents is not easy. They recommended the use of model diagrams in the document to provide a better picture of the model.

As the use of graphical representation has clear advantages over textual representation due to its abstraction powers, the recommendation looks quite convincing. However, due to the time limitations it can only be targeted for future release of the approach.

8.3.6 Solution as a whole

Questions 7, 8 and 9 in the questionnaire were formulated to get an idea about the overall usefulness of the approach as an editing and reviewing solution for process model’s textual data. Both the interviewees thought that the solution is far better than doing editing and reviewing through the use of previous facilities in Spearmint. The basic reasons they provided were the ability to use MSWord’s editing and reviewing functionalities and the re-importing of edited and reviewed documents.

8.3.7 Limitations of the solution

Last question in the questionnaire was not targeted to validate the usefulness of the approach. Rather it was used to get an advice on how to cater the problem of limited horizontal space in the document. This limitation was creating problems while representing multiple levels of sub-entity sections using indentation.
Both the interviewees said that the existence of more than 4 or 5 levels of sub-entities is quite uncommon practically. This factor, along with the advice to remove indentation for distinguishing sub-entity sections, helped to remove the limitation of only 2 sub-entity levels.
9. Conclusion

This thesis provided an approach for representing n-dimensional software process models in one-dimensional word processing documents.

The development of the approach was motivated by the fact that current software process modeling tools lack the capabilities of generating word processing documents that can keep model semantics for its computer process-ability and human understandability. This consequently results into inefficient use or nonuse of word processing tools, which otherwise can be very effective for editing and reviewing large amounts of text typically associated with a software process model.

The developed approach fills this gap by providing a way to generate word processing documents that are both computer process-able and human understandable. Computer process-ability was achieved by providing a way to annotate model’s textual data with semantically built XML tags, while human understandability was achieved by providing a mechanism for representing model constructs, such as entities and relationships, using word processor text layout and formatting features.

The approach is generic and can be adapted for use with a number of types of software process models. However, due to a number of limitations in using the approach for more than one word processing tool, the presented approach targets only MSWord 2003.

The approach is validated for its use to generate semantic word processing documents from models in a software process modeling tool. The outcomes of the validation show the practicality and usability of the approach.

9.1 Answers to the research questions and research contributions

The research in this thesis was aimed to answer some research questions. This section examines how the thesis answers those questions and in this context what are its contributions to software process modeling research.

Q1. What are the issues related to representing an n-dimensional software process model in a one-dimensional word processing document?

The thesis presented a set of seven issues that hinder the straight representation of n-dimensional software process models in one-dimensional word processing documents. These issues are mainly concerned with building human understandable and computer process-able semantics in word processing documents.

Providing all these issues in one place is one major contribution of the thesis, not only for the software process modeling domain but for the research of building
semantics in textual documents. This is because the issues are presented in a way that they can be used to guide any research for building semantics in textual documents.

**Q2. How can these issues be resolved in a solution for the representation problem?**

The development of the approach was guided by the issues for representing n-dimensional software process models in one-dimensional word processing documents. In fact the major aim while developing the approach was to resolve these issues.

The human understandability issues were resolved by providing a set of nine visual presentation rules, presented in Appendix I, while computer process-ability issues are resolved by providing ways to define and use XML tags.

By answering this question, the thesis contributes to the software process modeling research by providing an approach for representing n-dimensional software process models in one-dimensional word processing documents. In order to make this contribution more useful, a major aim while developing the approach was to achieve generality of the approach, so that it can be used with many different types of software process models.

**Q3. How can this solution be used by software process modeling tools to facilitate editing and reviewing of model’s textual data?**

This question was answered by implementing the approach in a software process modeling tool called Spearmint. The implementation of the approach helped validating its practicability, while its use to generate and re-import WordML documents provided a mechanism for editing and reviewing software process model’s textual data.

This mechanism can also be utilized to form a complete process model textual data review process. This can be done by following below steps:

1. Generate WordML document from software process model created in Spearmint.
2. Provide copies of generated document to software process model reviewers.
3. Get their feedback in the form of updated documents.
4. Use MSWord’s “Compare and Merge Documents” facility to combine all the reviewed copies in a single MSWord document.
5. Re-import the final document to Spearmint.

Although the usefulness of the approach to facilitate process model’s textual data editing and reviewing is validated to some extent, the usefulness of the implementation to facilitate the complete process model textual data review process
is not validated so far. This provides an important future work as discussed in chapter 10.

9.2 Lessons Learned

This section highlights some of the lessons learned during the research carried out in this thesis.

9.2.1 Building human understandable semantics is challenging

Research in the areas of semantic web, and knowledge management has focused a lot on building computer process-able documents. This resulted into many techniques for building computer process-able knowledge in documents, such as the use of XML tags to annotate document’s textual data. It has been observed that although these techniques make a document computer process-able, they do not guarantee their human understandability. For example XML documents are easy to interpret by computers but they are hard to edit by humans. This is complimented by the fact that human understandability varies from person to person. A concept that is very easy to understand by one person may not make much sense for another person.

9.2.2 Hierarchical relationships are easy to represent

While analyzing the issues for representing models in documents it was observed that despite the single dimension-ness of word processing documents they are very useful to represent hierarchical relationships. This can be achieved with the use of heading/sub-heading structures, which are normally very understandable for humans. Other facilities, such as document maps, ease the understanding of hierarchical relationships even further.

9.2.3 Existence of three steps for building semantic documents

A finding during the analysis of already available techniques for building semantics in documents was that despite the versatility of the research areas in which these techniques were developed, most of the techniques were based on three basic steps. These steps are 1) modeling real world concepts 2) building rules from concepts 3) generating semantic documents based on rules. It has been realized that these common steps can be used for the development of more generic techniques for building semantics in documents.
10. Future work

Following are the aims of the research ahead:

10.1 Release of next version of the approach

The initial work after the completion of the thesis will be to update the approach based on the dynamic validation results. The major proposed updates are presented in chapter 8.

10.2 Validation of complete review process

As discussed in section 9.1, the validation of how well the implemented solution facilitates the review process has not been performed so far. This involves the steps to distribute word processing documents (generated from Spearmint models) to reviewers, getting their feedback, merging the feedback into one document and re-importing the document back to model. Validating the implemented solution in this complete review process is one major future goal.

The reason why this validation was not performed as the part of the thesis is the unavailability of an industrial setting in which this scenario can be validated. However, search for organizations that are willing to participate has already been started, through the help of researchers at Fraunhofer IESE.

10.3 Release of the implemented solution for industry use

Once the results of the validation of complete review process in industry settings will be obtained, they will be used to update both the approach and its implementation in Spearmint. This solution will then be made available for industry use as a part of Spearmint toolset.
References


Appendix I - Visual presentation rules

This appendix presents a set of nine visual presentation rules that can be used to represent entities and relationships in a WordML document. These rules also take into consideration the relationship representation issues and provide solution to cater them.

Visual presentation rules use the notion of sections to structure the document. A section can be considered as a container of entities, properties, or even relationships. Each section has its own heading and description and it adds an extra level of hierarchy in the document. Four types of sections, which are used to discuss relationship representation issues, are entity section, property section, sub-entity section and association section. These section types along with their contents are described in visual presentation rules.

Each visual presentation rule is labeled with a numbered heading, followed by a short definition of the rule, followed by a detailed description.

**VP.R1:** Represent each entity as a text-block, with its own heading, description and ID.

As everything in a word processing document is text, an entity must also be represented in a textual form. This can be done with the use of entity text-blocks. Three basic properties of each entity text-block are its heading, description and ID. Heading and description can be used to make the text-block visually identifiable, while ID can be used to make it distinguishable for computers.

In addition to the basic properties, an entity text-block can also contain other entity specific characteristics. These additional characteristics depend on the meta-model created in Step 1 of the approach, and some of the visual presentation rules defined below.

**VP.R2:** Represent text-block heading using hierarchical-level number and name of entity.

The heading of each entity text-block consists of two parts, a number representing the hierarchical order of the text-block followed by the text representing the name of the entity. For the format of the heading either the word build-in heading styles or user defined styles can be used.

An example of a heading is “1.1.4 Elicit requirements”. Where 1.1.4 is the number part of the heading, which represents that the entity is at third level of hierarchy and it is sequentially fourth among its peers, at that level of hierarchy. “Elicit requirements” is the name of the entity.

**VP.R3:** Represent text-block description as plain text under its heading.
Another property of every text-block is its description. Description can be represented as plain text in the following paragraph of the heading. For example the description of the “Elicit requirements” entity section along with its heading can be represented as below:

1.1.4 Elicit requirements

Description of elicit requirements goes here.

**VP.R4**: Form an entity section corresponding to each non-abstract entity type in meta-model.

Generalization relationships are the type of relationships that exist only implicitly at model instance level and it is not very useful to represent them through the use of graphical or textual notations. This is complimented by the fact that some of the entity types can also be defined as abstract, i.e. their instances cannot be created. Keeping the constraints of abstract entity types in view, the generalization relationships shown in Figure 1 cannot be represented at model instance level, if *Resource* is defined as an abstract entity type. This is because representing them needs the existence of *Resource* instances at model instance level, which is not possible if *Resource* is an abstract type.

![Figure 1: Representing generalizations](image)

Although generalizations are not represented at model instance level they are used to define entity sections in this approach. This is done by defining an entity section for each non-abstract entity type in the meta-model. For example, if *Resource* is a type of entity with *Tool* and *Human* as its specializations. Then all three entity types can have a separate section in Word document, if the instances of all of them can be created. However, if the model restricts the instantiation of *Resource* entity type and instead allows creation of only its derived entity types then only the derived entity types will form sections. The example of this second case of document is shown below:

1. **Tools**
   - Description of tools goes here.
2. **Humans**
   - Description of humans goes here.
### VP.R5:

Represent an entity of a particular entity type in corresponding entity section.

The contents of each entity section are all the entity instances (or entity text-blocks) of that entity type, against which the entity section has been created. An example is:

1. **Tools**
   - Description of tools goes here.
   - **1.1 Eclipse**
     - Description about Eclipse goes here.

2. **Humans**
   - Description of humans goes here.
   - **2.1 Smith**
     - Description about Smith goes here.
   - **2.2 George**
     - Description about George goes here.

### VP.R6:

Use the rules defined in below table to represent properties.

<table>
<thead>
<tr>
<th>Nature of property</th>
<th>Representation rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-to-one non-textual area</td>
<td>Represent as attribute of entity. Viewable through Word 2003 custom schema facilities.</td>
</tr>
<tr>
<td>All other</td>
<td>Form a property section corresponding to each property. This section appears as sub-section of entity text-block and it contains a table incase of composite or one-to-many properties.</td>
</tr>
</tbody>
</table>

Representation of two of the property relationships – name and description – is already explained above. Both these relationships are special form of property relationships and thus are dealt separately. For representation of other property relationships, properties are divided into two parts. One part contains those properties that need to be represented with in the textual area of the document. The other part contains those properties that do not need to be represented in textual area of the document. The decision on which properties should appear in textual area is dependent on whether the property needs to be reviewed or not.

The properties that do not need to be represented in the textual area of the document can be represented as attributes of the entity in XML-Schema. Although Word 2003 does not show entity attributes in document’s textual area, it provides other ways to view them.

Above explained solution for representing non-textual area properties only works for one-to-one properties and not for one-to-many form of
non-textual are properties. Therefore, they will be dealt along with textual area properties.

A textual area property can be represented by forming a sub-section in entity’s text-block. If the property is a one-to-one property and its type is based on some basic data types, such as string, the property section can contain textual value in a paragraph. An example is:

1. **Tools**
   Description of tools goes here.
   1.1 **Eclipse**
   Description about Eclipse goes here.
   1.1.1 **Version**
   3.2.0

One-to-many form of properties or properties that are composed of many basic data types can be represented by putting a table in the property section. Each row of the table represents one instance of the property relationship, with first row used as header of the table. An example of a one-to-many composite property is:

1. **Tools**
   Description of tools goes here.
   1.1 **Eclipse**
   Description about Eclipse goes here.
   1.1.1 **AddIns**

<table>
<thead>
<tr>
<th>Add-In</th>
<th>Description</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ant build</td>
<td>Description of ant</td>
<td>1.1</td>
</tr>
<tr>
<td>Sub-version</td>
<td>Description of SV</td>
<td>2.1</td>
</tr>
</tbody>
</table>

**VP.R7**: Use the rules defined in below table to represent aggregations.

<table>
<thead>
<tr>
<th>Nature of aggregation</th>
<th>Representation rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contained type is equal to container’s type</td>
<td>Form a sub-entity section in container entity’s text-block and put all contained entity text-blocks init.</td>
</tr>
<tr>
<td>Contained type is not equal to container’s type</td>
<td>Keep entities in their respective entity sections and use association representation mechanism to represent their aggregation.</td>
</tr>
</tbody>
</table>

In order to define rules for representing aggregation relationships, it is necessary to divide them into two groups. One in which the contained entity’s type is same to the container’s type and the other in which the types are different. The former type can be represented by forming a sub-entity section for all the contained entities in container entity’s text-block. For example:
1. Activities
Description of activities goes here.

1.1 Elicit requirements
Description goes here.

1.1.1 Sub-Activities
1.1.1.1 Interview stakeholders
Description goes here.
1.1.1.2 Read available material
Description goes here.

The second group of aggregations, i.e. the aggregations in which the contained entities are different from container entity, cannot be represented in a sub-section of container entity. This is due to rule VP.R5, which states that entities of a particular entity type must be in the corresponding entity section.

Due to the impossibility of using document’s hierarchical structure to represent second group of aggregations, they can also be represented in a similar way as many-to-many aggregations. That is they can also be dealt as associations.

**VP.R8:** Create an association section under entity’s text-block and construct an association table in it. Use each row of the table to represent one type of associations. Represent association instances in each row using hyperlinks that link an entity’s text-block to its associated entity’s text-block.

An association in WordML document can be represented by using MS Word’s hyperlink mechanism, with each hyperlink representing one association relationship instance. In the directed associations the hyperlink only appears in the source entity’s text-block and it points to the destination entity’s text block. However, in undirected associations, the hyperlinks appear in both participating entity text-blocks, each pointing to the other text-block. Whether the association is directed or undirected the text of the hyperlink is the name of the target entity instance and it points to target entity heading.

All the association instances are placed into a sub-section of an entity’s text-block. This sub-section is named “Relationships” and it contains a table with each row representing one particular type of associations. First column of the table contains the type of the association and second column contains comma separated hyperlinks. An example of relationship table is presented below:

<table>
<thead>
<tr>
<th>Relationship type</th>
<th>Relationship instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Following/Activities</td>
<td>Create Design, Create</td>
</tr>
<tr>
<td></td>
<td>Test-cases</td>
</tr>
<tr>
<td>ProducedArtifacts</td>
<td>SRS</td>
</tr>
</tbody>
</table>
**VP.R9:** Use binary operators to build data/control flow conditions in association table rows.

Control and/or data flow conditions are normally divided into two categories, namely join conditions and split conditions. The join condition has multiple inflows and one outflow, where the condition decides which of the inflows must be true in order for the outflow to take place. Similarly, the split condition has one inflow and multiple outflows, where the condition decides which of the outflows must take place once the inflow becomes true. An inflow is considered true if its preceding processing unit has finished its task and false if it has not finished.

Representing control/data flows is similar to representing associations. However, the conditions need some extra formalism. This can be done by introducing binary operators, such as AND, OR, XOR, and brackets, such as ‘{}’, ‘()’, ‘[]’, to formulate a flow condition expression in a particular association table row. The split conditions can be showed in the source entity’s relationship table, while join conditions can be showed in the destination entity’s relationship table. An example is shown below:

1. **Activities**
   Description of activities goes here.
   
   1.1 **Design system architecture**
      Description goes here.
      
      **1.1.1 Relationships**
      
      | Relationship type | Relationship instances |
      |------------------|------------------------|
      | PrecedingActivities | (Create SRS AND Create Prototype) |
      | FollowingActivities | {(Detailed Design AND Create Test-cases) OR (System Development)} |
Appendix II – Questionnaire

Below questions were asked from the researchers in the area of software process modeling and reviews, to validate the usefulness of the approach for editing and reviewing software process model’s textual data. The questions are quite open ended and they were only used to initiate discussion on certain aspects of usefulness.

1. How do you like the representation of properties, i.e. name, description, attributes, etc?

2. Does the mechanism of representing hierarchical relationships help you understand the context of container/contained entities?

3. Does the mechanism of representing associations help you understand the context of associated entities?

4. Does the representation of flow conditions convey their meaning as modeled in the tool?

5. How do you like the model navigation mechanism through the use of hyperlinks?

6. Do you think this method of editing and reviewing process models textual data replaces the need to understand the model through graphical representation?

7. Does the way Spearmint modeling concepts are presented in the document convey their meaning or some mechanism to show the mapping of concepts from Spearmint to document is necessary along with the document?

8. As from your experience (experience from a model reviewing activity) you felt many problems to update document contained models that cannot be imported back to Spearmint. Do you think this mechanism of generating and re-importing model documents brings resolution to those problems? If yes then can you highlight some of them? If no then can you give suggestions for improvements?

9. Is the approach more suitable, less suitable, or unchanged to edit and review process textual data, rather than the old approach used in the tool?

10. What are the practical issues of limiting the sub-entity levels?