Chapter 1
Semantically Integrated Conceptual Modeling Method and Modeling Patterns

Remigijus Gustas
Karlstad University, Sweden

Prima Gustiene
Karlstad University, Sweden

ABSTRACT
Managing evolutionary changes, identification of discontinuities, and separation of concerns is not an easy task in the area conceptual modeling in information system development. One of the fundamental problems is that most conventional conceptual modeling techniques deal with the collection of loosely linked meta-models, which are defined by different types of diagrams. Typically, system development methods project interactive, behavioral, and structural aspects of information systems’ conceptual representations into disparate views. Therefore, the semantic integrity of various architecture dimensions is difficult to achieve. In this chapter, the authors present a semantically integrated conceptual modeling method. The advantage of this method is stability and flexibility of the diagrams to manage the constant changes of system requirements. This method provides the possibility to visualize the interplay among structural, interactive, and behavioral aspects. This is very important for the control of semantic integrity and to maintain a holistic representation where external and internal views of service conceptualizations are visualized together. Such visualization is also important for separation of concerns, which provides foundation for creation of modeling patterns. Modeling patterns are important for several reasons. First, they can be used for demonstration of the interplay of fundamental constructs that are used for system analysis and design. Secondly, modeling patterns are important for the evaluation of the expressive power of semantic modeling languages. It is demonstrated by case studies that sequential, underlying, enclosing, overriding, and overlaying interaction loops between actors provide the foundation for the composition of complex scenarios, which span across organizational and technical system boundaries.
INTRODUCTION

Every enterprise system can be seen as a composition of the organizational and technical components, which are viewed as various types of enterprise actors (Gustas & Gustiene, 2007). Although many requirements can be attributed to an individual component, still there are many requirements that impact many components. Such requirements cut across components and a called crosscutting concerns (Jacobson & Ng, 2005). Conventional information system (IS) analysis and design methods are restricted in their ability to distinguish among crosscutting concerns, which span across various types of diagrams. It does not matter whether designers apply structured analysis and design (SAD) methods (Gane & Sarson, 1979), (Yourdon & Constantine, 1979), object-oriented or component based methods (OMG, 2010): their expressive power is limited in separating various concerns. To break down a problem into smaller parts is called separation of concerns (Jacobson & Ng, 2005). Disability to manage separation of concerns is one of the reasons why the way systems are currently built is rather primitive and meet a lot of problems. Consequently, managing the complexity of specifications in software engineering is the problem that can be attributed to various limitations of traditional IS modeling and design methods. To obtain value from the graphical representations they must be integrated and semantically correct.

In the traditional areas of engineering, developers are able to present their design decisions by using a finalized computation-neutral representation. This is not a case in the area of system engineering. The limitations of conventional system modeling methods result in two side effects, which in aspect-oriented software development (Jacobson & Ng, 2005) are known as tangling and scattering. Tangling occurs when the software component or class, instead of fulfilling a particular concern, encapsulates a diverse set of concerns. If a particular concern is spread across multiple components, then this situation is called scattering. When the requirements caused by that concern are modified, the designer must identify all related components and to find out how these components are affected by introduced changes. Especially, modifying requirements, which are related to a big number of diagrams, become quite problematic. Poor understanding of concerns makes it difficult to make even simple evolutionary extensions of IS specifications. Separation of crosscutting concerns (Gustas & Gustiene, 2012) is the first fundamental problem, which cannot be solved without modifying modeling foundation in system analysis and design. In this paper, we introduce a new way of modeling and decomposition principles, which suggest a new and more natural way of managing complexity in system engineering. We also present four modeling patterns that we constructed using this semantically integrated conceptual modeling method (SICM).

The declarative nature of value flow exchanges help technical system designers to analyze underlying business events, which are quite comprehensible for such stakeholders as business process modeling experts, enterprise architects, and users. Diagnosing value flows among different organizational components in IS engineering is important for solving the alignment problem (Wieringa, 2008), (Wieringa & Gordijn, 2005) of value models (Gordijn & Akkermans, 2000) with the behavioral effects and structural changes in various classes of objects. Value exchanges and related coordinating events can be used as the guidance for system designers to move smoothly from system analysis to design, without a requirement to represent a complete solution. By sending and receiving value flows, the actors enter into commitments regarding their privileges, rights, responsibilities and obligations. One of the reasons why the conventional system analysis and design methods are not suitable for modeling the deontic aspects of organizations (Wagner, 2003), such as commitments and claims (Chopra et al., 2010), is that they not able to capture value and coordi-