Domain Engineering – Using Domain Concepts to Guide Software Design

Iris Reinhartz-Berger  
Department of Management Information Systems, University of Haifa, Haifa 31905, Israel  
iris@mis.hevra.haifa.ac.il

Arnon Sturm  
Department of Information System Engineering, Ben-Gurion University of the Negev, Beer Sheva 84105, Israel  
sturm@bgumail.bgu.ac.il

Yair Wand  
Department of Management Information Systems, University of Haifa, Haifa 31905, Israel  
wand@mis.hevra.haifa.ac.il

Abstract:
As the variability of information and software systems has increased, the need for an engineering discipline concerned with building reusable assets (such as specification sets, patterns and components) on one hand and representing and managing knowledge in specific domains on the other hand has become crucial. This discipline, called domain engineering, supports the notion of a domain, defined as a set of applications that use a set of common concepts for describing requirements, problems and capabilities. The purpose of domain engineering is to identify, model, construct, catalog, and disseminate a set of software artifacts that can be applied to existing and future software in a particular application domain. As such, it can support the effective and efficient management and development of software assets. Hence, it is important to introduce this discipline among software engineering practitioners and researchers.

As an evolutionary approach, domain engineering relates to many different areas, such as conceptualization, ontologies and ontology deployment, metadata and metamodeling, knowledge base integration, reuse, patterns, model driven approaches, and reference modelling.

The purpose of this tutorial is to present and discuss domain engineering concepts, methods, problems, and solutions. We will focus on modelling domains and applications and validating application models against their domain models. We will review several domain engineering methods, explaining their basics, rationale, advantages, and shortcomings. We will compare and discuss these methods using a case study, emphasizing how they support the reuse and knowledge management activities.

What is Domain Engineering?
Domain engineering is a discipline that supports the notion of a domain. A domain can be defined as a set of applications that use common concepts for describing requirements, problems and capabilities. The purpose of domain engineering is to identify, model, construct, catalog, and disseminate a set of software artifacts that can be applied to existing and future software systems in a particular application domain. Similarly to software engineering, domain engineering includes three main activities: domain analysis, domain design, and domain implementation. Domain analysis identifies a domain and captures its ontology. It should specify the basic elements of the domain, identify the relationships among these elements, and represent this understanding in a useful way. Domain design and domain implementation are concerned with mechanisms for translating requirements into systems that are made up of components with the intent of reusing them to the highest extent possible. Domain modelling is especially important because of two main reasons. First, modelling, especially visual modelling, can help understanding complex system
specifications and support communications among the various stakeholders engaged in the development process. Secondly, the core elements of a domain and the relationships among them usually remain unchanged, while the technologies and implementation environments are in progressive improvement. Hence, domain models usually remain valid for long periods.

Many methods and techniques have been developed to support domain modelling. FODA and PLUS, for example, are feature-oriented approaches that emphasize the common and different features of applications in a specific domain. GME, metaEdit+, and DOME are examples of metamodeling environments which enable definition of domain specific languages (DSL) and support using them to describe particular applications.

Nowadays, domain engineering and domain specific languages receive special attention from communities which deal with conceptualization, ontologies and ontology deployment, metadata and metamodeling, knowledge base integration, reuse, patterns, model driven approaches, and reference modelling. However, domain engineering has been criticized as dealing with too broad areas (domains) which are usually understood only during the development process. Most of these problems can be solved by taking a special care of the way domain engineering is woven into software engineering.

In this tutorial, we will present and discuss domain engineering concepts, problems, and solutions through the Application-based Domain Modelling (ADOM) approach. ADOM binds domain and application models into a general framework and enables the definition of mutual constraints between these types of models. An example of a domain can be process control systems which monitor and control the values of certain variables through a set of components that work together to achieve a common objective or purpose. Application areas within this domain include engineering and industrial control systems, control systems in the human body, and financial derivation-tracking products. All of these application types share concepts such as controller, controlled value, executer, sensor, etc. In addition, they should obey constraints, such as "keep the system within the boundaries of the controlled value."

Being influenced by Meta Object Facility (MOF), the architecture of ADOM consists of three layers: the application layer, the domain layer, and the (modeling) language layer. The application layer consists of models of particular applications, including their structure (scheme) and behaviour, e.g., a home climate control system and a financial derivation-tracking product. The language layer comprises metamodels of modeling languages, e.g., UML. The intermediate domain layer consists of specifications of various domains, e.g., the process control systems domain. The ADOM architecture also enables enforcing constraints among the different layers; in particular the domain layer enforces constraints on the application layer. The fulfilment of these constraints can be checked due to the ability to classify application elements according to the domain terminology.

**Tutorial Objectives:**
The main purpose of the tutorial is to present and discuss domain engineering concepts and methods, focusing on domain modelling activities. This can be divided into the following objectives:

- Introducing basic concepts of domain engineering.
- Reviewing and comparing common domain modeling methods.
- Discussing domain engineering problems and solutions to these problems.
• Exemplifying domain engineering concepts and their integration into software engineering processes through the Application-based Domain Modeling (ADOM) approach.

**Detailed Outline:**

**Introduction (30 minutes)**
- The landscape of software reuse and knowledge representation and management (15 minutes)
- What is domain engineering: concepts and problems (15 minutes)

**Overview of domain modelling approaches (120 minutes)**
- Common domain engineering methods, their basics and rationales (60 minutes).
- The Application-based Domain Analysis Modelling (ADOM) Approach (60 minutes)

**Summary and discussion (30 minutes)**

**Intended Audience and Required Background:**
The tutorial is targeted to researchers and practitioners in different software engineering areas.
The tutorial is at the introductory level, but requires basic background in object-oriented analysis and design (preferably in UML).
Biography of the Presenters:

**Iris Reinhartz-Berger** is a faculty member at the Department of Management Information Systems, Haifa University, Israel. She is also adjunct faculty at the Technion - Israel Institute of Technology. She received her PhD in Information Management Engineering from the Technion, Israel Institute of Technology. Her research interests include conceptual modeling, modeling languages and techniques for analysis and design, domain analysis, and systems development processes. She has published her work in journals and international conferences. In addition, Iris has 7 years of teaching experience, including software engineering, methodologies, analysis, design, and domain engineering courses and seminars.

**Arnon Sturm** is a faculty member at Ben-Gurion University. He is also adjunct faculty at the Technion - Israel Institute of Technology and Haifa University. Arnon has 9 years of teaching experience, including software engineering courses. His research concentrates on the topics of the proposed tutorial. Arnon has published several papers on these topics. Prior to his studies, Arnon has gained extensive experience in developing software systems in the industry. He served as a member of a software engineering team that addressed problems similar to those addressed by the proposed tutorial. Arnon has also given several tutorials at international scientific meetings, including EASSS 2003, EASSS2004, and AAMAS 2003.

**Yair Wand** is CANFOR Professor of MIS at the Sauder School of Business, the University of British Columbia, Canada, and a Professor at the MIS Department, Faculty of Social Sciences, Haifa University, Israel. He received his D.Sc. in Operations Research from The Technion (Israel Institute of Technology), his M.Sc. in Physics from the Weizmann Institute (Israel), and B.Sc. in Physics from the Hebrew University, Jerusalem. His current research interests include theoretical foundations for information systems analysis and design, development and evaluation of systems analysis methods, and conceptual modelling. In particular, he has done work on the use of ontological concepts in information systems analysis and design. Presently he is on the editorial boards of the *Journal of the Association of Information Systems (JAIS)*, *Journal of Data Semantics, Journal of Database Management, Requirements Engineering Journal* and *Applied Ontology*.

Contact information:
Dr. Iris Reinhartz-Berger
Department of Management Information Systems
Haifa University
31905 Israel
Tel: 972-4-828-8502
Fax: 972-4-828-8522
Email: iris@mis.hevra.haifa.ac.il