Preface

Domain Engineering is an engineering discipline concerned with building reusable assets, such as specification sets, patterns, and components, in specific domains. A domain in this context can be defined as an area of knowledge that uses common concepts for describing phenomena, requirements, problems, capabilities, and solutions. The purpose of domain engineering is to identify, model, construct, catalog, and disseminate artifacts that represent the commonalities and differences within a domain. Although being applicable to different engineering disciplines, domain engineering methods and domain specific languages (DSL) receive nowadays special attention from the information systems and software engineering researchers and practitioners who deal with artifact reuse, application validation, and domain knowledge representation. In particular, these topics are of interest in the areas of software product line engineering and ontology engineering. One of the reasons for this interest is the increasing variability of information and software systems and the need to obtain and share expertise in different, evolving domains.

Domain engineering deals with two main layers: the *domain layer*, which deals with the representation of domain elements, and the *application layer*, which deals with the software applications and information systems artifacts. In other words, the programs, applications, or systems are included in the application layer, whereas their common and variable characteristics, as can be described, for example, by patterns or emerging standards, are generalized and presented in the domain layer.

Similarly to application engineering, domain engineering includes three main activities: domain analysis, domain design, and domain implementation. Domain analysis identifies a domain and captures its ontology. Its aim is to specify the basic concepts of the domain, identify the possible relationships among these concepts, and represent this understanding in a useful way. Domain design and domain implementation are concerned with mechanisms for translating requirements to artifacts that will operate in the domain, i.e., into systems that are made up of components with the intent of reusing them to the highest extent possible. All these activities are performed within the domain layer. However, domain engineering also supports inter-layer activities, namely interactions that exist between the domain and application layers. Specifically, domain layer artifacts may be reused and used for the design and validation of the specifications of application layer artifacts, while the applications may be generalized into domain artifacts in a process that can be termed knowledge elicitation. Figure 1 visually summarizes the two layer model of domain engineering and its related activities.

Domain engineering as a discipline has practical significance as it can provide methods and techniques that may help reduce time-to-market, product cost, and projects risks on one hand, and help improve product quality and performance on a consistent basis on the other hand.

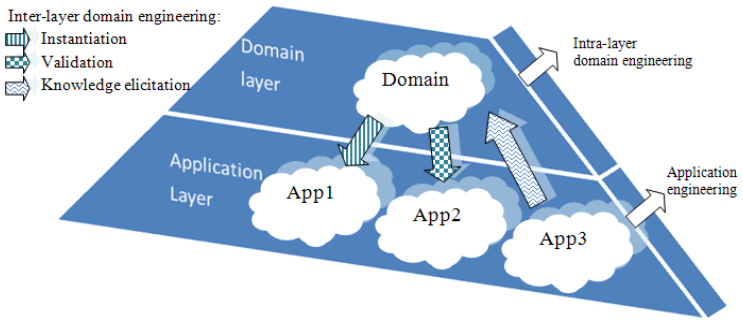

Figure 1. The two layer model of domain engineering

The purpose of this workshop is to bring together researchers and practitioners in the area of domain engineering in order to identify possible points of synergy, common problems and solutions, and visions for the future of the area. The workshop accepted 7 papers and 1 invited talk in the following topics:

Invited talk:

Jorn Bettin, Model Oriented Domain Analysis: an Industry Voice

Modeling approaches:

1. Niklas Mellegård and Miroslaw Staron, A Domain Specific Modelling Language for Specifying and Visualizing Requirements.
2. Catherine S. Price, Jules-Raymond Tapamo, Felicity Blakeway, and Fethi Ahmed, Plantation Forestry: an Analysis of the Domain.
3. Iris Reinhartz-Berger, Domain Aspects: Weaving Aspect Families to Domain-Specific Applications

Supporting tools and frameworks:

4. Catalin Constantin, Vincent Englebert, and Philippe Thiran, A Reconciliation Framework to Support Cooperative Work with DSML.
5. Ruben Heradio and David Fernandez, Towards a time-efficient algorithm to calculate the total number of products of a Software Product Line.

Analyzing specific domains:

6. Abdelouahed Gherbi, Pejman Salehi, Ferhat Khendek, and Abdelwahab Hamou-Lhadj, Capturing and Formalizing SAF Availability Management Framework Configuration Requirements
7. Camelia Maga and Nasser Jazdi, Concept of a Domain Repository for Industrial Automation

Iris Reinhartz-Berger, Arnon Sturm, and Yair Wand
DE@CAiSE'2009 co-chairs

For more information on the workshop, see our website http://www.bgu.ac.il/~sturm/DE@CAiSE09/, or contact Iris Reinhartz-Berger (iris@mis.haifa.ac.il), Arnon Sturm (sturm@bgu.ac.il)

Organization

DE@CAiSE'09 co-chairs

Dr. Iris Reinhartz-Berger	Dr. Arnon Sturm	Prof. Yair Wand,
University of Haifa, Israel	Ben Gurion University of the Negev, Israel	University of British Columbia, Canada

Program committee

Colin Atkinson	University of Mannheim, Germany
Mira Balaban	Ben-Gurion University of the Negev, Israel
Sholom Cohen	CMU-SEI, USA
Kim Dae-Kyoo	Oakland University, USA
Dov Dori	Technion – Israel Institute of Technology, Israel
Joerg Evermann	Memorial University of Newfoundland, Canada
Jeff Gray	University of Alabama at Birmingham, USA
Atzmon Hen-Tov	Pontis, Israel
Steven Kelly	MetaCase, Finland
Philippe Kruchten	University of British Columbia, Canada
John McGregor	Clemson University, USA
Dirk Muthig	Fraunhofer Institute for Experimental Software Engineering, Germany
Klaus Pohl	University of Duisburg-Essen, Germany
Iris Reinhartz-Berger	University of Haifa, Israel
Michael Rosemann	The University of Queensland, Australia
Julia Rubin	IBM Haifa Research Labs, Israel
Bernhard Rumpe	Braunschweig University of Technology, Germany
Lior Schachter	Pontis, Israel
Klaus Schmid	University of Hildesheim, Germany
Keng Siau	University of Nebraska-Lincoln, USA
Pnina Soffer	University of Haifa, Israel
Il-Yeol Song	Drexel University, USA
Arnon Sturm	Ben Gurion University of the Negev, Israel
Giancarlo Succi	Free University of Bozen-Bolzano, Italy
Juha-Pekka Tolvanen	MetaCase, Finland
Yair Wand	University of British Columbia, Canada
Gabi Zodik	IBM Haifa Research Labs, Israel

Additional reviewers

Dirk Reiss, Braunschweig University of Technology, Germany
Holger Eichelberger, University of Hildesheim, Germany