1. Introduction

Domain-specific languages [1, 2, 3] are programming languages for solving problems in a particular domain and provide built-in abstractions and notations for that domain. Domain-specific languages are usually small, more declarative than imperative, less expressive and more attractive than general-purpose languages because of easier programming, systematic reuse, better productivity, reliability, maintainability, and flexibility. However, the benefits of domain-specific languages are not for free. The cost of domain-specific language design, development and maintenance has to be taken into account. Without an appropriate methodology and tools these costs can be higher than savings obtained by the later use of domain-specific languages. Advantages of the formal definitions of general-purpose languages should be exploited, taking into consideration the special nature of domain-specific languages. To be productive, the development of these languages has to be based on high-level automated tools [4].

The purpose of this mini-track is to bring together an international audience of researchers and practitioners actively involved in the development of domain-specific languages that support the software engineering process. It covers a wide range of domain-specific languages applied in different software engineering problems. Some topics of interest for this mini-track are:

- design and implementation of domain-specific languages,
- use of domain-specific languages in the software engineering process,
- the role of domain-specific languages in software engineering,
- tools, environments, and techniques needed to support domain-specific languages.

In “A language for Software Subsystem Composition” by Jim Buffenbarger and Kirk Gruell, a domain-specific language for specifying the composition and construction of a software system is described. The authors clearly show how to design and implement a domain-specific language and what are the benefits of its usage.

Next, Walter A. Risi, Pablo E. Martinez Lopez and Daniel H. Marcos in “HyCom: A Domain-Specific Language for Hypermedia Application Development” describe a domain-specific language for hypermedia applications. Hypermedia applications are now in growing interests and the paper shows how the design and implementation of such applications benefit if a domain-specific language is used to specify the design. A hypermedia application is finally obtained by an automatic application generator.

Finally, Peter Pfahler and Uwe Kastens in “Configuring component-based specifications for domain-specific languages” describe the development of domain-specific languages using domain-specific components, which consist of the implementation part (specifications to compiler generator) and the interface part (specifications of dependencies between components). The domain-specific language is then described as a composition of domain-specific components. Since domain-specific components can be configurable, a whole family of domain-specific languages can be built. Using the Jacob tool a domain expert can build a particular domain-specific language by selection. The approach is interesting and differs from other domain-specific language development approaches.

2. References