SOFISMO
Your data pilot

FORMAL SOFTWARE ENGINEERING

Formal software engineering starts with formal requirements engineering. Nowadays, this is achieved on the one hand with formal process definitions and on the other hand with domain-specific modelling (DSM) at all abstraction levels down to the individual artefact. The basic idea behind formal requirements engineering with DSM is to provide every domain expert (technical as well as business) with a custom-made modelling or specification language which he uses to define the desired functionality in the domain he is responsible for. These languages are modular by design and completely independent of implementation technology. They constitute the foundation for automated transformations into a large variety of derived artefacts such as documentation, scripts, configuration parameters and – of course – program code. With DSM the requirements engineering process reaches unprecedented levels of productivity and reproducibility. In fact, the DSM methodology could have a major influence on the survival of software development in high-wage countries like Switzerland and Germany.

Summary of the methodology called "Domain-Specific Modelling"

Domain-specific modelling (DSM) originated in a discipline called "Product Line Engineering" which is more than 10 years old. It is based on a complete separation of the concerns „specification" (aka: requirements engineering) and "implementation".

Requirements Engineering

- The first step towards formal requirements engineering is a domain analysis resulting in the modularisation of the problem description. An interview process focussing on the owners of deep domain knowledge and structured by a dozen pre-defined questions is used to divide the problem domain into a set of small sub-domains.
- Each sub-domain is then modelled with a domain-specific modelling language (DSML). Such a language is defined by the concepts (aka: abstractions) that domain experts use to describe requirements with respect to a particular domain. The higher the level of abstraction of a DSML, the better it allows users to express complex requirements in compact models that are easily understood by other domain experts.
- Subsequently, users of a DSML agree on one (or more) notation(s) that represent(s) the underlying concepts. As far as possible, these notations should reflect tried and true nomenclatures and symbols that domain experts have been using in traditional specification documents. In a large organisation it is not uncommon to find several notations (so-called jargons) for a single DSML. Notations can be textual or graphical and are usually embedded in universal editing frameworks (e.g. Xtext for textual notations) that support the comfortable and efficient manipulation of model instances. Thus, users neither need to use unknown general-purpose tools nor do they need to express themselves in unfamiliar languages such as UML.

- The resulting specification is a set of model instances that is complete, unambiguous and free of redundancies. All changes are modelled only once and automatically propagated throughout all the derived artefacts.

Implementation
In principle, a specification defined with DSM can be implemented with traditional methodologies. However, since the specifications are defined in a formal language, the transformation to fully functional applications can easily be automated.

- All the domain-specific modelling languages (DSML) are implemented via a reference application containing at least one instance of each language concept used in the DSMLs. This implementation is done by hand with the technologies and products that are currently used by the organisation. The resulting implementation (source code, configuration files etc.) is then partitioned into a set of parameterized implementation templates. These templates represent a generalised mapping function between domain terminology and implementation technology. The implementation may contain an arbitrary mix of new and old technologies as well as commonly used off-the-shelf products or custom-made frameworks.
- Once all business and technical sub-domains have been mapped onto the relevant implementation technologies, any model instance can automatically be translated into working software via a model-driven generator that assembles the corresponding set of implementation templates (software factory). Such a development process leads to unprecedented levels of agility and productivity, and reduces the total cost of ownership of bespoke software systems dramatically.

Advantages of Domain-Specific Modelling (DSM)
(from the point of view of requirements engineering)

Domain-specific modelling is applicable to any task where complex decisions have to be documented in a reproducible manner. For an IT-development organisation the following advantages are relevant:

- Since DSM is based on a set of small specification tools defined by the users themselves, any domain expert can participate directly in the requirements-engineering activity.
- A given problem can be partitioned into small sub-problems along the structure of the underlying organisation.
- Multiple specification artefacts such as diagrams, concepts, messages, drawings, models, etc. can be synchronised to form a consistent specification for subsequent implementations.
- Due to its modular structure, DSM can be introduced in small steps along the structures of an organisation. Roll-out into production is achievable incrementally, without any big-bang introduction of new tools across the entire organisation.
- At any stage of the introduction of DSM, the organisation can continue with more traditional approaches and still benefit from the results already achieved.
- The whole specification process is technology-independent. Specifications created with DSM can be implemented on any technology.
- All the tools needed for DSM, can be found in the Open-Source community. The latest releases of Eclipse contain a very good set of such tools.
- The costs for the development and the maintenance of bespoke software systems drop by at least 30%.
- **DSM brings 100% traceability to the development of software systems and a nearly perfect alignment of business and IT.**
- **DSM makes the core know-how of business entities explicit and independent of technology.**