

# Free/Open Model Development – Towards Liberating End-Users

Janne Merilinna and Katja Henttonen  
VTT Technical Research Centre of Finland  
P.O. Box 1000, 02044 Espoo, Finland  
[[janne.merilinna](mailto:janne.merilinna@vtt.fi), [katja.henttonen](mailto:katja.henttonen@vtt.fi)][@vtt.fi](mailto:janne.merilinna@vtt.fi)

## 1. Introduction

Nowadays many digital devices, such as set-top boxes and mobile phones, provide an Application Programming Interface (API) which allows power users to tailor the accompanying software. For example, one can make changes to an unfit user interface to increase usability of a device or build custom extensions to introduce new features. There are also communities which share such modifications and extensions as free or open source software (F/OSS). Laymen (i.e. people who have no programming skills) can benefit by downloading and installing these ready-made software bundles. However, usability and a need for custom features depend on personal preference and many laymen would like to tailor the software by themselves.

The question remains on how to enable computer illiterate people to modify and extend the software on their digital devices. Third-generation programming languages (3GLs) cannot be used for this purpose, not even with the most user-friendly API imaginable. However, it may be possible for laymen to program with Domain-Specific Modelling Languages (DSMLs).

This article introduces a revolutionary approach which enables laymen to become contributors in the F/OSS movement. The approach is based on a combination of Domain-Specific Modelling (DSM) and F/OSS development. Herein, we call it "Free or Open Models" (F/OM) development. In F/OM, users share (a) visual models from which complete code generation is possible and (b) a modelling infrastructure for various application domains. Both the models and the infrastructure can be used, modified and extended freely.

The rest of this article is as follows. First, DSM is introduced in order to set background for F/OM. Second, F/OM is presented. Conclusions close the article.

## 2. Domain-Specific Modelling

Model-Driven Development (MDD) is about treating models as first class design entities. Inside MDD, at least two different approaches exist: general purpose

modelling and domain-specific modelling. General purpose modelling languages, such as UML, can be used for wide a variety of purposes across a broad range of domains. Elements of a programming language (e.g. methods, parameters and objects) are used to model a solution for a problem. These modelling elements are only understandable to programmers. In contrast, DSMLs use modelling elements which are known to all practitioners in a specific domain. For example, in the domain of home security automation systems, modelling elements include e.g., light switches, security cameras, electrical door locks, motion sensors. The behaviour of the system is modelled by combining these elements according to certain domain-specific rules.

Because DSMLs utilize concepts and rules of the problem-space instead of concepts of solution-space, application developers don't need to know a programming language or platform specific details in an optimal case. DSML also takes advantage of complete code generation, i.e. executable source code is automatically generated from domain-specific models. Industry cases constantly experience 5-10 productivity gains when using DSMLs compared to using traditional software development means [1]. Further, there is evidence that application development can be made so easy that even non-programmers can do it (such as demonstrated by a DSML for an insurance product case) [1].

However, these benefits don't come from void. In order to enable rapid and easy application development, someone has to first implement the DSM basic architecture. The basic architecture consists of a modelling language (i.e. a metamodel), code generators and domain-specific software framework. DSM also requires a language workbench, i.e. a tooling environment for modelling applications and developing modelling languages and code generators. There are F/OSS or gratis environments available such as Generic Eclipse Modeling System (GEMS)<sup>1</sup> and GME (Generic Modeling Environment)<sup>2</sup>. There are also proprietary industry-ready DSM environments such as MetaEdit++.

---

<sup>1</sup> <http://www.eclipse.org/gmt/gems/>

<sup>2</sup> <http://www.isis.vanderbilt.edu/projects/gme/>

### 3. Free or Open Models Software Development Approach

F/OM is an approach for developing free applications with DSMLs. The principal idea of F/OM is the same as it is in F/OSS; develop applications collaboratively. The essential difference is in the amount of contributors. In F/OSS laymen can act either as passive end-users or bug reporters but in F/OM they can also become active developers.

#### 3.1. Collaboration

Essentially, two kinds of collaboration work should take place in the F/OM community: (1) applications development and (2) DSM basic architecture development. These tasks are explained briefly in the following.

In F/OM *applications* development, models are primary assets being shared instead of source code. One can develop an application from scratch by using a provided DSML and then share the application, i.e. a domain-specific model(s), with others. The others can tailor and enhance the application model and give their improvements back to the community. The application development can be made so easy that even programming illiterate people can participate, thus being able to enjoy all software freedoms<sup>3</sup> for the first time.

However, more experienced core developers must collaboratively develop and maintain the underlying DSM basic architecture, i.e. DSMLs, generators, and frameworks. The DSML embodies the domain-specific modelling elements, interrelations, constraints and notations. The software framework is an optional element and is mainly for abstracting low-level platform details and gathering commonalities in single location whereas the DSML defines the variability space. The generators are always specific to a meta-model and underlying software framework. Multiple generators might be needed to enable porting the same application models to different target platforms.

#### 3.2. F/OM Collaboration Platform

To make F/OM a reality, an environment for developing DSM basic architecture and applications must exist. Firstly, a language workbench is required. While advanced developers would enjoy using existing DSM tooling environments (e.g. GEMS), there should also be a simple, online modelling environment for laymen. There is thus a need to develop both a

common, web-based modelling environment and adaptors for utilizing existing DSM workbenches.

Secondly, there must be a platform for collaborative development of free models. Essentially, the F/OM Collaboration Platform would provide project maintenance features similar to SourceForge or Savannah but also take the special characteristics of DSM into account. Figure 1 illustrates a sketch of such a platform. Each F/OM project contains the application models and the DSM basic architecture. The web-based DSM language workbench and adaptors would be developed in a separate Tools project.

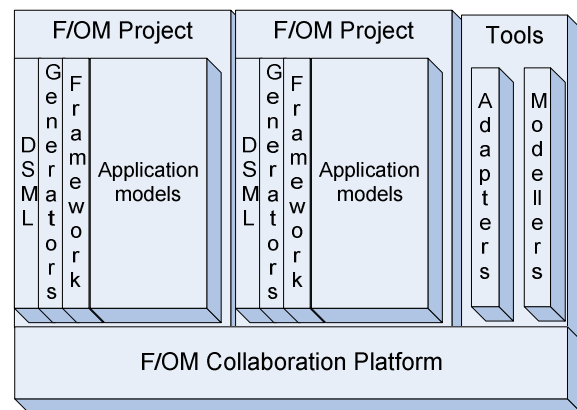


Figure 1. F/OM Collaboration Platform.

### 4. Conclusion

Usability of different devices is always a matter of user's preferences. Therefore, to satisfy the various and divergent needs of end-users, they should be enabled to modify the provided software. Previously, laymen end-users were not able to program with 3GLs, but they could be enabled to do so with DSMLs. In this article, such an approach was introduced. We call the approach F/OM. To make F/OM reality, preliminary high-level conceptual architecture for F/OM Collaboration Platform was introduced. If such a platform was developed, we anticipate that it would be a directed revolution for increasing end-user satisfaction with various digital devices.

### 5. References

- [1] Kelly, S. and Tolvanen, J-P, Domain-Specific Modeling – Enabling full code generation, John Wiley & Sons, New Jersey, 2008, 427p., ISBN: 978-0-470-03666-2.
- [2] Roberts, D., Johnson, R., Evolving Frameworks: A Pattern Language for Developing Object-Oriented Frameworks, Proceedings of Pattern Languages of Programs, 1996

<sup>3</sup> <http://www.gnu.org/philosophy/free-sw.html>