FAMA Framework*

Pablo Trinidad David Benavides Antonio Ruiz-Cortés Sergio Segura
Alberto Jimenez
University of Seville
Av. Reina Mercedes s/n, 41012 Seville, Spain

Abstract

FAMA Framework (FAMA FW) is a tool for the automated analysis of variability models (VM). Its main objective is providing an extensible framework where current research on VM automated analysis might be developed and easily integrated into a final product. FAMA FW is built following the SPL paradigm supporting different variability metamodels, reasoners or solvers, analysis questions and reasoner selectors, easing the production of customized VM analysis tools. FAMA FW is written in Java and distributed under Apache License.

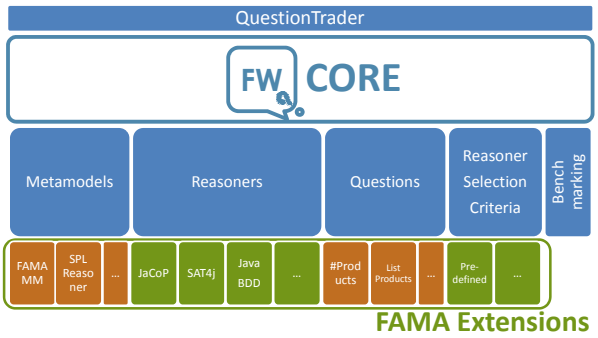

Figure 1. FAMA Framework Architecture

1 Context

In [3] we presented *FAMA Eclipse plug in* as a tool for the automated analysis of feature models that integrated three logic paradigms and their respective solvers: Constraint Solver Programming (CSP) by means of JaCoP, Propositional Satisfiability (SAT) by means of SAT4j and Binary Decision Diagram (BDD) by means of JavaBDD. Using this tool, you may perform different reasoning operations on feature models, like calculating the number of products in a Software Product Line (SPL), getting a list of its products, filtering products according to a criterion or detecting and explaining errors.

FAMA FW arises with the intention of allowing third parties to integrate their automated reasoning techniques into a workspace where some basic features are provided by default. Among these features we may find VM file storing, a benchmarking system to compare the performance of several reasoners and VM random generation.

FAMA FW is built following SPL paradigm so a customer may build a customized distribution only with some desirable features. Furthermore, its component-based architecture allows to extend or update FAMA FW by means of the so-called *FAMA Extensions*. We summarize the benefits of using FAMA FW:

- *Easy to use*: FAMA FW has a simple and stable front-end (*QuestionTrader*), implementing a query-based interaction.
- *Easy to extend*: FAMA FW architecture allows to extend or update existing products just by adding or updating its components or features. Third-parties are allowed to develop and integrated their own FAMA Extensions.
- *Easy to configure*: FAMA FW is configured by means of an unique XML file, easing its maintenance and configuration to adapt the tool to the user needs.

2 FAMA FW Architecture

From our experience on feature model analysis [1] we have performed a domain analysis process with the aim of detecting the common features of different analysis tools. Those features and the different extension points are part of the FAMA FW Core. The remaining features are provided as FAMA Extensions.

In Figure 2 you may see a detailed description of FAMA FW architecture that we briefly describe to understand the

*This work has been partially supported by the European Commission (FEDER) and Spanish Government under CICYT project Web-Factories (TIN2006-00472)

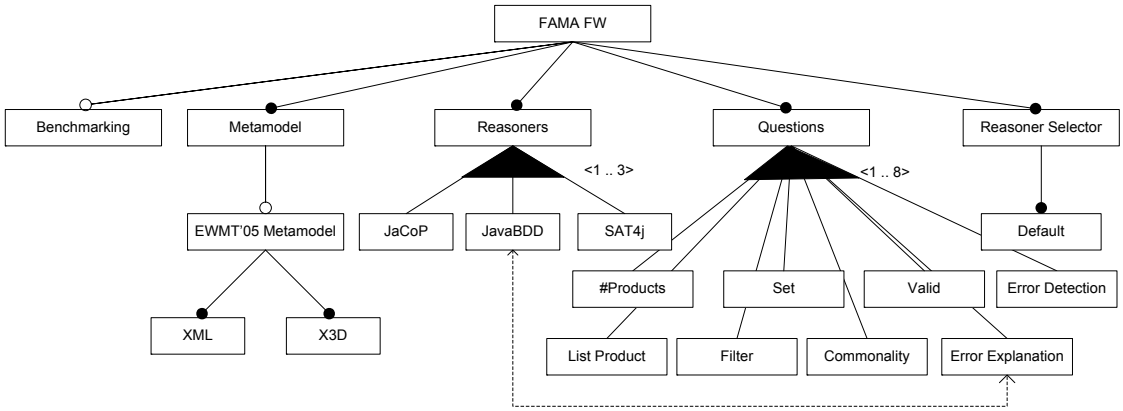

Figure 2. FAMA FW Feature Model

functionality and the flexibility of the tool:

- *QuestionTrader*: interfaces for a query-like interaction with FAMA FW. They are uncoupled from FAMA Core so changes on it does not affect the final user.
- *FAMA Core*: communicates different extensions among them.
- *FAMA Extensions*: set of modules that allows FAMA FW to adapt to user needs:
  - *Metamodels*: integrate your own variability or feature metamodel and file loaders/savers into FAMA FW. FAMA FW supports FAMA Eclipse plug in metamodel and XML and X3D (3D representation of feature models) file formats by default.
  - *Reasoners*: add new reasoners or solvers developed by third parties. By default, JavaBDD, JaCoP and SAT4j are provided with the tool.
  - *Questions*: incorporate new reasoning operations [2]. Products counting and listing, error detection and explanation [5], and products filter are included to date.
  - *Reasoner Selector*: select at runtime the reasoner to answer an analysis question among all the available reasoners and based on a previously defined criterion (performance, accuracy, ...). FAMA FW allows to develop your own smart selectors.
  - *Benchmarking*: compare different reasoners performance for the same operation. A random feature model generator is provided allowing to store the data and results for a later analysis.

FAMA FW is distributed as a set of self-contained *Jar* files under Apache 2.0 license. Its available at `http:://www.isa.us.es/fama` where you may additionally find the related papers and companies and institutions that are currently using the tool.

3 Future work

We are currently working on integrating existing metamodels such as extended feature models (those containing extra-functional attributes) into FAMA FW. Feature model refactoring and merging [4] are being developed as new supported questions. New reasoners will be developed to integrate other CSP reasoners such as Choco. Finally, we are very interested in FAMA FW users feedback so we are able to build an agenda to develop a product that really fits into user needs.

References

[1] D. Batory, D. Benavides, and A. Ruiz-Cortés. Automated analysis of feature models: Challenges ahead. *Communications of the ACM*, 49(12):45–47, 2006.

[2] D. Benavides, A. Ruiz-Cortés, P. Trinidad, and S. Segura. A survey on the automated analyses of feture models. In *Jornadas de Ingeniería del Software y Bases de Datos (JISBD)*, 2006.

[3] D. Benavides, S. Segura, P. Trinidad, and A. Ruiz-Cortés. FAMA: Tooling a framework for the automated analysis of feature models. In *Proceeding of the First International Workshop on Variability Modelling of Software-intensive Systems (VAMOS)*, pages 129–134, 2007.

[4] S. Segura, D. Benavides, A. Ruiz-Cortés, and P. Trinidad. Automated merging of feature models using graph transformations. *LNCS*, to be determined, 2008.

[5] P. Trinidad, D. Benavides, A. Durán, A. Ruiz-Cortés, and M. Toro. Automated error analysis for the agilization of feature modeling. *Journal of Systems and Software*, 81(6):883–896, 2008.