QUANTICOL

A Quantitative Approach to Management and Design of Collective and Adaptive Behaviours

quantiCOL

http://www.quanticol.eu

TR-QC-2-2013

Chaining available tools to support the modelling and analysis of a bike-sharing product line

An experience report

Revision: 1; Nov 15, 2013

Author(s): Maurice H. ter Beek (CNR–ISTI), Stefania Gnesi (CNR–ISTI), and Alessandro Fantechi (UNIFI)

Publication date: Dec 20, 2013

Funding Scheme: Small or medium scale focused research project (STREP)
Topic: ICT-2011 9.10: FET-Proactive 'Fundamentals of Collective Adaptive Systems' (FOCAS)
Project number: 600708

Coordinator: Jane Hillston (UEDIN)
e-mail: Jane.Hillston@ed.ac.uk
Fax: +44 131 651 1426

Part. no.	Participant organisation name	Acronym	Country
1 (Coord.)	University of Edinburgh	UEDIN	UK
2	Consiglio Nazionale delle Ricerche – Istituto di Scienza e Tecnologie della Informazione "A. Faedo"	CNR	Italy
3	Ludwig-Maximilians-Universität München	LMU	Germany
4	Ecole Polytechnique Fédérale de Lausanne	EPFL	Switzerland
5	IMT Lucca	IMT	Italy
6	University of Southampton	SOTON	UK

COOPERATION

Contents

Abstract

Bike-sharing systems are becoming popular not only as a sustainable means of transportation in the urban environment, but also as a challenging case study that presents interesting run-time optimization problems. As a side-study within a research project aimed at quantitative analysis that used such a case study, we have observed how the deployed systems enjoy a wide variety of different features. We have therefore applied variability analysis to define a family of bike-sharing systems, and we have sought support in available tools. We have so established a tool chain that includes (academic) tools that provide different functionalities regarding the analysis of software product lines, from feature modelling to product derivation, to quantitative evaluation of the attributes of products. The tool chain is currently experimented inside the cited project as a complement to more sophisticated product-based analysis techniques.

1 Introduction

Bike-sharing systems (BSS) are becoming popular not only as a sustainable means of transportation in the urban environment, but also as a challenging case study that presents interesting run-time optimization problems.

A side-study of the recently started EU project QUANTICOL (`www.quanticol.eu`) concerns the quantitative analysis of BSS seen as collective adaptive systems (CAS). The design of CAS must be supported by a powerful and well-founded framework for quantitative modelling and analysis. CAS consist of a large number of spatially distributed entities, which may be competing for shared resources even when collaborating to reach common goals. The nature of CAS, together with the importance of the societal goals they address, mean that it is imperative to carry out thorough analyses of their design to investigate all aspects of their behaviour before they are put into operation. In this context it is important to realize that the behaviour of the individual entities from which a CAS is composed, may exhibit variability not only in the kind of features but also in the quantitative characteristics of features themselves.

Starting form the BSS case study identified in QUANTICOL, we have started to apply variability analyses on a bike-sharing product line that we have defined. First we have sought support in available tools for the possibility of adding attributes and quantitative characteristics to our BSS specification. This has resulted in the tool chain that we present in this paper. It includes (academic) tools that provide different functionalities regarding the analysis of software product lines, from feature modelling to product derivation, to quantitative evaluation of the attributes of products. The tool chain is currently being experimented further inside QUANTICOL as a complement to more sophisticated product-based analysis techniques.

As far as we know, there was no study available concerning the possible different realizations of a BSS starting from its description as a product line and then using methods and tools developed in the field of software product lines to (i) analyze the different admissible products by looking at the possible variabilities and, moreover, (ii) taking into account the different attributes that may be used to measure, e.g., the development cost of the various derivable products.

The paper is organized as follows. In § 2 we introduce the bike-sharing case study. In § 3 we show how to model the BSS with SPLOT, after which we show how to automatically translate such models into a format suitable for FeatureIDE in § 4. In § 5 we instead show how to automatically translate such models into a format suitable for ClaferMOOVisualizer and how to moreover add attributes. To conclude, the lessons we learned from this experience are presented in § 6.

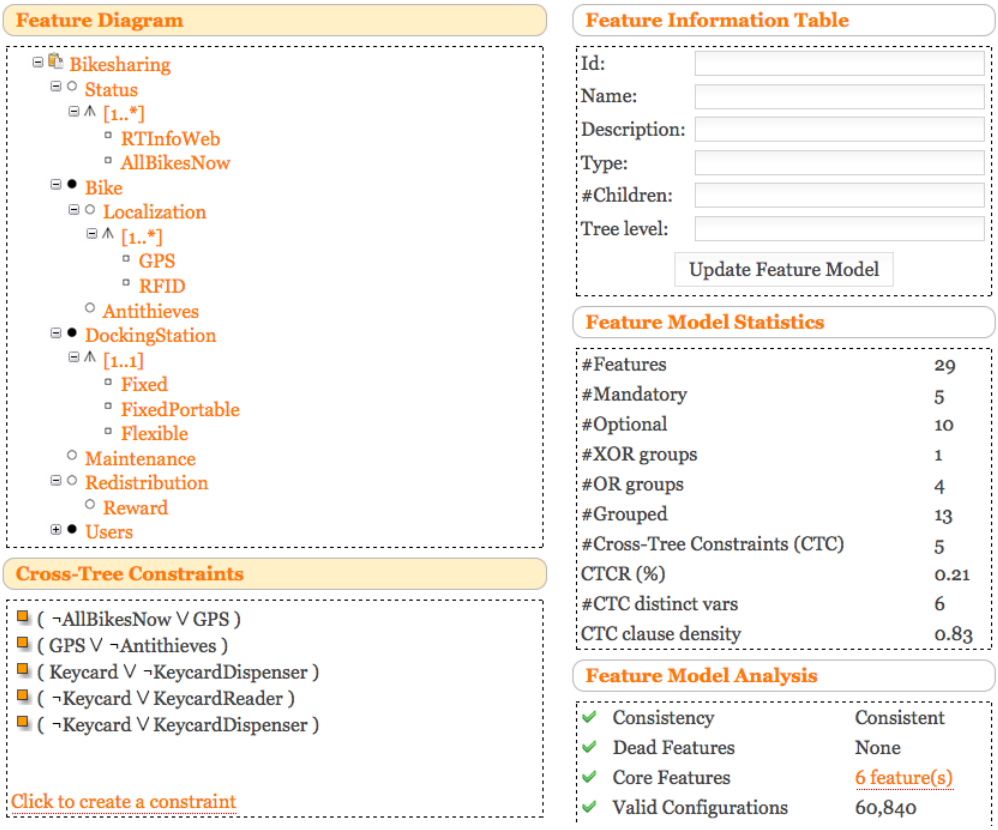

Figure 1: The BSS Feature Model in S.P.L.O.T., not showing the Users feature's subfeatures

2 BSS: Bike-Sharing Systems

An increasing number of large, medium and small size cities worldwide are adopting fully automated public bike-sharing systems (BSS) as a green urban mode of transportation [2]. The concept is simple (a user arrives at a station, pays for a bike, uses it for a while and returns it to a station) and their benefits multiple, including the reduction of vehicular traffic (congestion), pollution, and energy consumption.

The current third generation technology-based BSS have almost nothing (but the bikes) in common with the first generation free BSS introduced in Amsterdam nearly half a century ago. *Vélib′*, the well-known and highly successful BSS of the city of Paris, currently consists of over 20,000 bikes and some 1,800 stations. There are now similar BSS in more than 500 cities worldwide. The largest BSS can be found in China with 50,000 bikes and 2,000 stations, one every 100 meters. Fourth generation BSS are already being developed. These include movable and solar-powered stations, electric bikes and mobile (i)phone real-time availability applications [5].

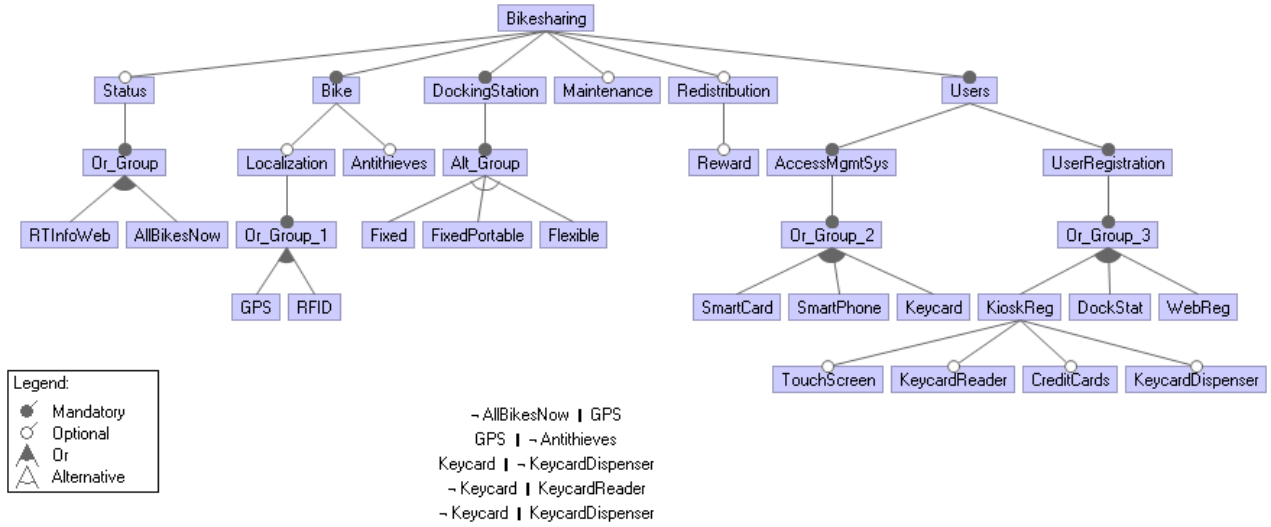

Figure 2: The complete BSS Feature Model in FeatureIDE

In the context of QUANTICOL we are collaborating with "PisaMo azienda per la mobilità s.p.a.", an in-house public mobility company of the city of Pisa's administration. They recently introduced the public BSS *CicloPi* in the city of Pisa, which currently consists of 200 bikes and 12 stations.

More in detail, a BSS consists of parking stations distributed over a city, typically in close proximity to other public transportation hubs such as subway and tram stations. (Subscribed) users may rent an available bike and drop it off at any station in the city. To improve the efficiency and the user satisfaction of BSS, the load between the different stations may be balanced, e.g., by using incentive schemes that may change the behaviour of users but also by efficient (dynamic) redistribution of bikes between stations.

The deployment of such different measures gives rise to several interesting multi-objective optimization problems, which make this case study an interesting benchmark for the QUANTICOL project.

3 Modelling a BSS with S.P.L.O.T.

In order to apply variability analysis techniques to the case study, we developed an initial feature model by taking into account the main characteristics (features) of BSS as described in the literature (mainly [5]). These include bikes equipped with an optional localization feature (RFID or GPS) and an optional antithieves feature (which requires GPS), parking stations that have either a fixed permanent capacity or a fixed portable capacity or a flexible capacity, optional maintenance and redistribution of bikes, and – finally – an optional scheme of incentives based on rewards. Obviously we could have taken many more characteristics of BSSs into account, but we believe that the chosen ones represent a sufficient starting point for our exploratory study.

The resulting feature model representation in Fig. 1 was created with S.P.L.O.T.'s feature model editor, which is an online application developed by Marcílio Mendonça and others at the University of Waterloo [4]. Software Product Lines Online Tools is actually a web portal which integrates a number of research tools. S.P.L.O.T. allows to edit, debug, analyze, configure, share and download feature models (its feature model repository currently has nearly 400 entries). In particular, it allows to save models online to consult them later or to export them in the SXFM format.[1] However, it does not allow code generation, nor does it provide a means to create a graphical representation of a feature model.

[1]SXFM stands for Simple XML Feature Model, a concise textual format to denote feature models.

4 From S.P.L.O.T. to FeatureIDE

A tool that does allow to directly generate code (Java or C++) as well as a graphical representation from a feature model is FeatureIDE [8]. FeatureIDE actually supports the full lifecycle of a software product line, from domain engineering to feature-oriented software development. However, it accepts feature models in an XML format. To nevertheless be able to use FeatureIDE to work with feature models created with S.P.L.O.T. (or directly with one of the feature models in its repository), it thus becomes necessary to interpret the SXFM format. To this aim, we have developed a program that automatically converts SXFM files into the XML format accepted by FeatureIDE.

4.1 FMT: Feature Model Translator

We had to face two minor problems when we considered to write an automatic translation from SXFM to XML files:

1. Contrary to FeatureIDE, in S.P.L.O.T. the root of a group of subfeatures is itself not a feature, but rather a node in the feature diagram without a specific name.

2. Contrary to FeatureIDE, S.P.L.O.T. allows a feature model to have multi-features (a.k.a. as clones): features that appear several times in the same feature diagram (in fact, they can appear several times in a single product). This is actually dealt with by assigning a unique ID to each feature. In FeatureIDE, on the other hand, the name of a feature must unequivocally identify a feature (moreover, it cannot contain white spaces or any non-Latin symbols other than '_').

We resolved the first problem by simply associating a different name ('Or_Group_i', 'Or_Group_j', 'Alt_Group_i', etc., thus creating unique names) to each node of the feature model that identifies a group of subfeatures and by subsequently transforming it into a proper feature itself.

The second problem was resolved by eliminating all non-Latin symbols, by substituting all white spaces with '_'s and by subsequently simply adding a unique identifier to every occurrence of a multi-feature (e.g., multifeature_1, multifeature_2, etc.). Further modifications to feature names must be performed manually.

FMT is written in Java (using Swing's libraries for the GUI) in Eclipse. This choice allows FMT to run on many operating systems, including all those on which FeatureIDE can run, which is written in Java as well.

The actual translation of a feature model specified in the SXFM format to one in XML format consists of two phases:

1. While reading the SXFM file, the model is created in memory.

2. Starting from the model in memory, the XML file is written.

The choice for storing a model in memory has at least one advantage over a direct translation between two files: It eliminates the need to write a full translation for any set of two different file formats, as it may suffice to write the code for either reading a third format or writing a third format, thus improving reusability.

Providing FMT with the SXFM representation of the feature model depicted in Fig. 1, it thus generates an XML file that can be read by FeatureIDE. Using FeatureIDE's visualization functionalities we subsequently obtained the graphical representation of this feature model depicted in Fig. 2.

From this representation, FeatureIDE allows to generate configurations (i.e., products), an example of which is depicted in Fig. 3.

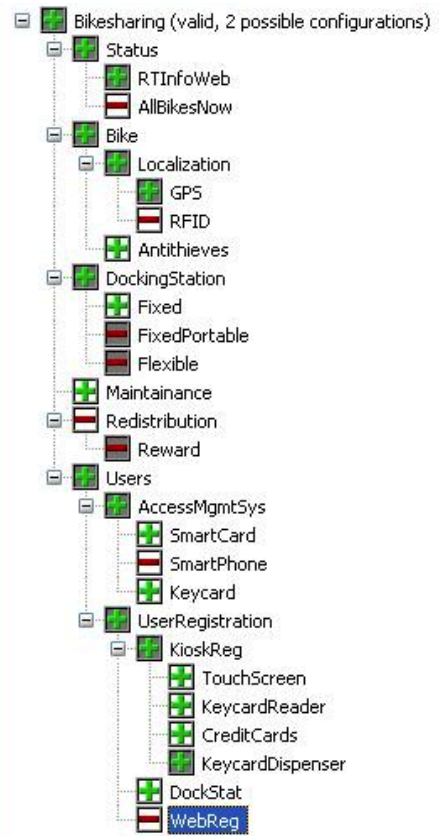

Figure 3: Possible BSS configurations

5 From S.P.L.O.T. to ClaferMOO

Until now we only considered ordinary feature models, i.e., feature diagrams modelling the hierarchical parent-child relationships between a set of features as a rooted tree and possibly some additional cross-tree constraints.

As outlined in the Introduction, in the context of QUANTICOL we are specifically interested also in quantitative analyses of BSS, in the sense that we consider the behaviour of the components of a BSS to exhibit variability not only in the kind of features that they possess, but also in the quantitative (non-functional) characteristics of their features.

To achieve this, we can add attributes and quantitative constraints among attributes and features to our BSS specification and consequently perform quantitative analyses. In other words, we consider the modelling and analysis of attributed feature models.

Neither S.P.L.O.T. nor FeatureIDE currently cater for attributed feature models, but FeatureIDE is being extended to support quality attributes [8].

5.1 Adding Attributes to the BSS

Clafer, a lightweight textual modelling language for software (product lines) developed jointly at the University of Waterloo and the IT University of Copenhagen, does specifically allow for attributed feature modelling [1]. Furthermore, splot2clafer, a small tool written in Java, automatically translates files from S.P.L.O.T.'s SXFM format into the CFR format of Clafer.

In Clafer, each feature can have an associated attribute and quality constraints can then be specified either globally or within the context of a feature. Think, e.g., of associating a cost to each feature and a global constraint that only allows products (feature configurations) whose total costs remain within a predefined threshold value.

This is an example of a single optimization objective, but usually there can be more than one attribute associated to a feature, leading to multiple optimization objectives. It suffices to imagine that each feature also has a value for user satisfaction associated to it and while the objective might be to minimize the cost of a product it might at the same time be desirable to maximize user satisfaction.

The ClaferMOO extension of Clafer was specifically introduced to support attributed feature models as well as the resulting complex multi-objective optimization goals [6]. A multi-objective optimization problem has a set of solutions, known as the Pareto front, that represents the trade-offs between two or more conflicting objectives. Intuitively, a Pareto-optimal solution is thus such that no objective can be improved without worsening another objective. A set of Pareto-optimal variants generated by ClaferMOO can be visualized (as a multi-dimensional space of optimal variants) and explored in the interactive tool ClaferMOOVisualizer, which was specifically designed to support software product line scenarios.

ClaferMOOVisualizer can help to understand the differences among variants, to establish their positioning with respect to various quality dimensions, to select the most desirable variants, possibly by resolving trade-offs, and – finally – to understand the impact that changes made during a product line's evolution have on a variant's quality dimensions.

We thus decided to annotate the features of the BSS with attributes and to define some global quantitative constraints over these attributes. For now we limited ourselves to the cost and customer satisfaction of the features and, in specific cases, their capacity and security; consequently, the global constraints aim to minimize the total cost of a configuration and to maximize customer satisfaction, capacity and security of a BSS.

The specification (in Clafer's CFR format) of the resulting attributed feature model is as follows:[2]

```
abstract Feature
    customersat : integer
    cost : integer
    capacity : integer

abstract SecurityFeature : Feature
   security : integer

abstract BIKES
    or Status : Feature ?
        [ customersat = 25 ]
        [ cost = 0 ]
        [ capacity = 0 ]
        RTInfoWeb : Feature
            [ customersat = 10 ]
            [ cost = 5 ]
            [ capacity = 0 ]
        AllBikesNow : Feature
            [ customersat = 20 ]
            [ cost = 10 ]
            [ capacity = 0 ]
    Bike : Feature
        [ customersat = 0 ]
        [ cost = 0 ]
        [ capacity = 0 ]
```

[2]The specification excludes the Maintenance, Redistribution and Users features as well as their subfeatures, as ClaferMOOVisualizer currently cannot handle such a large model.

```
        or Localization : Feature ?
            [ customersat = 3 ]
            [ cost = 3 ]
            [ capacity = 0 ]
            RFID : Feature
                [ customersat = 10 ]
                [ cost = 10 ]
                [ capacity = 0 ]
            GPS : Feature
                [ customersat = 15 ]
                [ cost = 15 ]
                [ capacity = 0 ]
        Antithieves : SecurityFeature ?
            [ customersat = 5 ]
            [ cost = 7 ]
            [ capacity = 0 ]
            [ security = 1 ]
    xor DockingStation : SecurityFeature
        [ customersat = 0 ]
        [ cost = 0 ]
        [ capacity = 0 ]
        [ security = 1 ]
        Fixed : Feature
            [ customersat = 17 ]
            [ cost = 30 ]
            [ capacity = 5 ]
        FixedPortable: Feature
            [ customersat = 20 ]
            [ cost = 35 ]
            [ capacity = 5 ]
        Flexible: Feature
            [ customersat = 23 ]
            [ cost = 40 ]
            [ capacity = 10 ]
    [ Antithieves => GPS ]
    [ AllBikesNow => GPS ]

    total_customersat : integer =
      sum Feature.customersat
    total_cost : integer =
      sum Feature.cost
    total_capacity : integer =
      sum Feature.capacity
    total_security : integer =
      sum SecurityFeature.security

Mybike : BIKES
<< max Mybike.total_customersat >>
<< min Mybike.total_cost >>
<< max Mybike.total_capacity >>
<< max Mybike.total_security >>
```

In Fig. 4, we see the result of optimizing this specification with ClaferMOOVisualizer. We see, e.g., that variant 11 offers maximal capacity and security at an affordable cost and with a reasonable customer satisfaction.

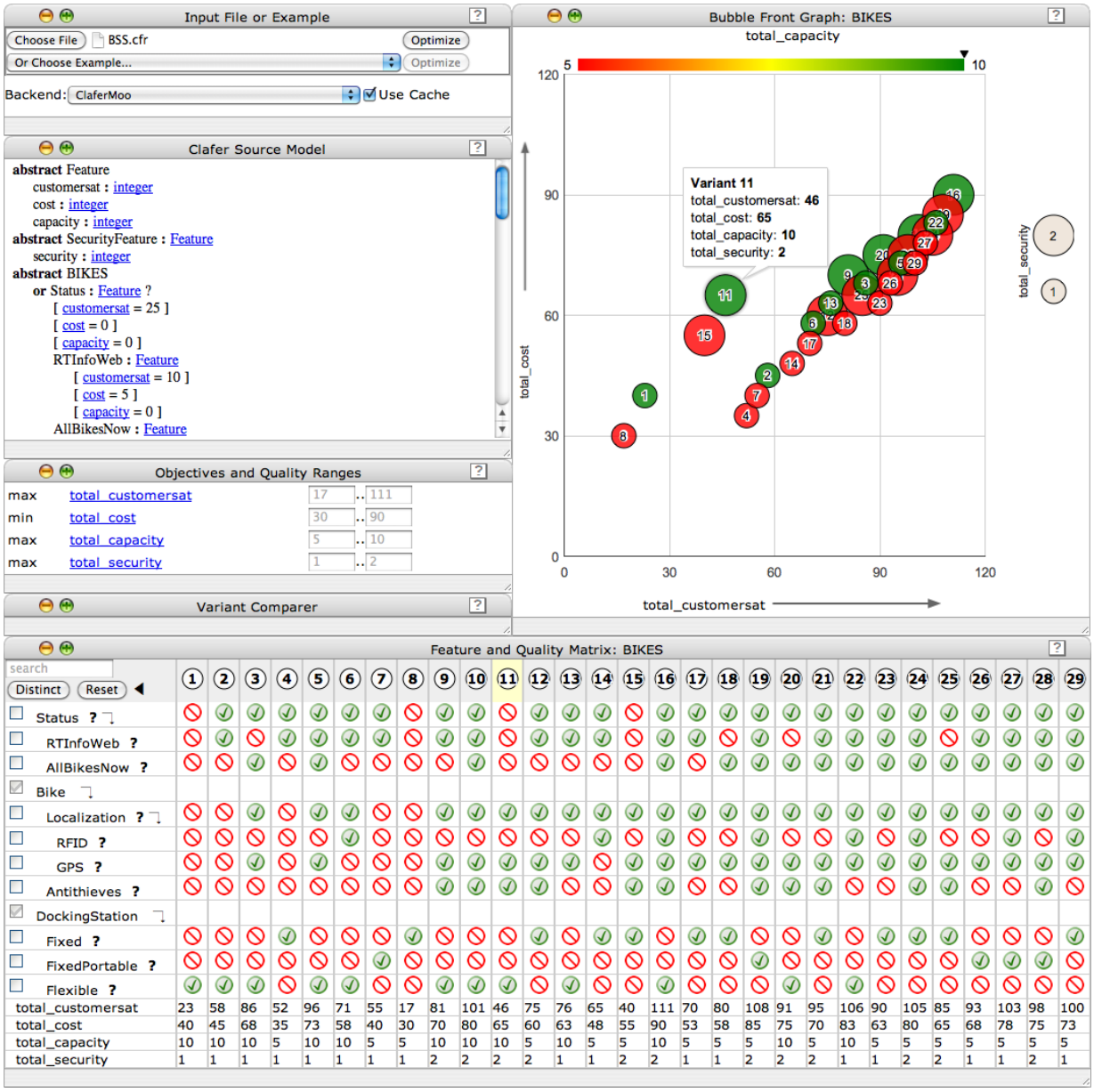

Figure 4: The BSS Feature Model in ClaferMOOVisualizer, corresponding to the BSS.cfr specification, i.e., excluding the Maintenance, Redistribution and Users features and their subfeatures

6 Lessons Learned

The main goal of the activity presented in this paper was to quickly set up and analyze a family of BSS, as a base for conducting quantitative evaluation of the different possible characteristics. We therefore chose to adopt existing feature modelling tools rather than to construct yet another modelling tool. In our analysis of academic, freely available tools, we soon realized that no single tool was ready to fully satisfy our expectations. It turned out that our best option was the synergic use of the chain of tools depicted in Fig. 5.

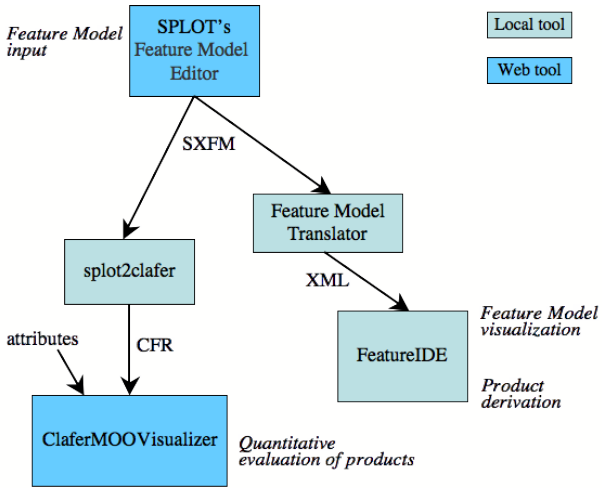

Figure 5: The toolchain experimented in this paper

Both S.P.L.O.T. and FeatureIDE provide the key functioning of what can be expected from a typical feature modelling tool: Creating, editing and analyzing a feature model, providing some statistics of the feature model, and deriving product configurations. A careful account, based on user experiences, of the commonalities and variability of the two tools is presented in [7], which confirms the fact that neither of the two is better than the other in all circumstances.

S.P.L.O.T. is a Web-based tool, including a repository of hundreds of feature models. It is quite user friendly and immediate to use, thanks to the availability of previously developed models. However, adding the model to the aforementioned repository is mandatory, which raises concerns over the privacy of the developed models that can thus be accessed and modified by anyone. Moreover, S.P.L.O.T. allows only a single product configuration to be created, which however cannot be saved in the repository.

FeatureIDE is a more traditional, locally executable tool, integrated in Eclipse. Hence, it allows not only the generation of products as feature combinations, but it also allows to automatically generate code skeletons that reflect the feature structure of a product within Eclipse itself. Although we did not exploit this feature in this paper, we consider it important for our intended future work on the BSS case study within QUANTICOL.

The complementarity of the two tools with respect to the above issues has been exploited in the experience described in this paper by first defining the feature model in S.P.L.O.T. and subsequently transferring it to FeatureIDE. This step has moreover been automated with a purposely built format transformation program.

Finally, we adopted ClaferMOOVisualizer for quantitative analyses of an attributed feature model, since – as far as we know – it is the only tool that exhibits this functionality. A specific tool for interfacing with S.P.L.O.T. exists, and this has been one of the reasons for maintaining a copy of the feature model in S.P.L.O.T.

Our experience with the tools was influenced by the fact that all of them are academic tools that at times present some minor problems: S.P.L.O.T. and FeatureIDE showed more maturity in this respect, while the online version of ClaferMOOVisualizer, possibly because it is the most recently developed of the three, still manifest some instability.

Obviously, having a single standard format for feature models, and maybe using locally running versions of the tools, would increase the synergy between the tools. For now, however, the exploited tool chain was sufficient for quickly modelling and analysing a bike-sharing product line.

The reported experience, although at a preliminary stage, has shown the general value of product line modelling when addressing a class of complex systems, given its ability to identify commonalities and variabilities between elements of the class. The added possibility of basing quantitative evaluation and optimization techniques on product line modelling is also appearing as a promising opportunity.

7 Acknowledgements

Maurice ter Beek and Stefania Gnesi are supported by the EU FP7-ICT FET-Proactive project QUANTICOL (600708) and the Italian MIUR project CINA (PRIN 2010LHT4KM).

The authors would like to thank Andrea Scozzarro for the implementation of FMT.

The authors would also like to thank Marco Bertini from PisaMo s.p.a. for kindly sharing his expertise on the public BSS *CicloPi* with us.

References

[1] K. Bąk, K. Czarnecki, and A. Wąsowski. Feature and Meta-Models in Clafer: Mixed, Specialized, and Coupled. In *Revised Selected Papers of the 3rd International Conference on Software Language Engineering (SLE'10)* (B.A. Malloy, S. Staab, and M. van den Brand, eds.). *Lecture Notes in Computer Science* 6563, Springer, 2010, 102–122.

[2] P. DeMaio. Bike-sharing: History, Impacts, Models of Provision, and Future. *Journal of Public Transportation* 12, 4 (2009), 41–56.

[3] C. Fricker and N. Gast. Incentives and Redistribution in Bike-Sharing Systems with Stations of Finite Capacity. arXiv:1201.1178v3 [nlin.AO], September 2013.

[4] M. Mendonça, M. Branco, and D.D. Cowan. S.P.L.O.T.: Software Product Lines Online Tools. In *Companion Proceedings of the 24th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'09)* (S. Arora and G.T. Leavens, eds.). ACM, 2009, 761–762.

[5] P. Midgley. Bicycle-Sharing Schemes: Enhancing Sustainable Mobility in Urban Areas. Background Paper CSD19/2011/BP8, Commission on Sustainable Development, United Nations Department of Economic and Social Affairs, May 2011.

[6] A. Murashkin, M. Antkiewicz, D. Rayside, and K. Czarnecki. Visualization and Exploration of Optimal Variants in Product Line Engineering. In *Proceedings of the 17th International Software Product Line Conference (SPLC'13)* (T. Kishi, S. Jarzabek, and S. Gnesi, eds.). ACM, 2013, 111–115.

[7] J.A. Pereira, C. Souza, E. Figueiredo, R. Abílio, G. Vale, and H.A.X. Costa. Software Variability Management: An Exploratory Study with Two Feature Modeling Tools. In *Proceedings of the 7th Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS'13)*, 2013, 36–45.

[8] T. Thüm, C. Kästner, F. Benduhn, J. Meinicke, G. Saake, and T. Leich. FeatureIDE: An Extensible Framework for Feature-Oriented Software Development. To appear in *Science of Computer Programming* 79 (2014), 70–85.