Applying the Product Lines Paradigm to the Quantitative Analysis of Collective Adaptive Systems

Maurice H. ter Beek
ISTI–CNR, Pisa, Italy
terbeek@isti.cnr.it

Alessandro Fantechi
DSI, University of Florence
and ISTI–CNR, Pisa, Italy
fantechi@dsi.unifi.it

Stefania Gnesi
ISTI–CNR, Pisa, Italy
gnesi@isti.cnr.it

ABSTRACT

Engineering a Collective Adaptive System (CAS) requires the support of a framework for quantitative modeling and analysis of the system. In order to jointly address variability and quantitative analysis, we apply the Product Lines paradigm, considered at the level of system engineering, to a case study of the European project QUANTICOL, by first defining a reference feature model and then adding feature attributes and global quantitative constraints, in the form of a Clafer attributed feature model. ClaferMOOVisualizer is subsequently used for quantitative analyses and multi-objective optimization of the resulting attributed feature model.

CCS Concepts

•Computing methodologies → Model verification and validation; •Software and its engineering → Software product lines;

Keywords

Collective Adaptive Systems, multi-objective optimization, ClaferMOO, quantitative modeling, quantitative analysis, variability analysis

1. INTRODUCTION

Collective Adaptive Systems (CAS) consist of a large number of spatially distributed entities, which may be competing for shared resources even when collaborating to reach common goals [11]. CAS are assuming an increasing societal importance, e.g., in the context of urban mobility, so that CAS engineering should include a comprehensive analysis of their design and investigate all aspects of their behavior before they are put into operation. In particular, CAS designs may exhibit variability in both the kind of features they offer and their quantitative characteristics. Engineering a CAS thus requires the support of a framework for quantitative modeling and analysis of the system design.

SPLC 2015, July 20 - 24, 2015, Nashville, TN, USA

ISBN 978-1-4503-3613-0/15/07…$15.00

DOI: http://dx.doi.org/10.1145/2791060.2791100

On the other hand, product line engineering has been proposed as an effective means to complexity management. Regarding CAS, managing variability consists in identifying variation points in a (collective) family design and deciding which combinations are valid products (individual pieces).

To jointly address variability and quantitative analysis, in this paper we apply the product lines paradigm, considered at the level of system engineering, by first defining a reference feature model and then adding feature attributes and global quantitative constraints, in the form of a Clafer attributed feature model. Consequently, ClaferMOOVisualizer is used for quantitative analyses and multi-objective optimization of the resulting attributed feature model.

We apply these principles to a case study of the European project QUANTICOL [5, 11] (www.quanticol.eu) concerning the quantitative analysis of bike-sharing systems (BSS) seen as an example of CAS.

The paper is organized as follows. In § 2 we introduce the bike-sharing CAS case study and we model the bike-sharing CAS as a product line, expressing it as a feature model, as also introduced in [6]. In § 3 we use Clafer to add quantitative attributes to features, and ClaferMOOVisualizer to perform a multi-objective quantitative analysis of the system. In § 4 we compare the manual variability management used in a classical design process with the variability management assisted by modeling and visualization tools.

2. AN EXAMPLE CAS: 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 [8]. The concept is simple (a user arrives at a docking station, rents a bike, uses it for a while and returns it to a station) and their benefits multiple, including the reduction of vehicular traffic, pollution, and energy consumption.

Next generation BSS are already being developed, including movable and solar-powered stations, electric bikes and smartphone real-time availability applications [14].

More in detail, a BSS consists of (docking) 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 behavior of users but also by efficient (dynamic) redistribution of bikes between stations.

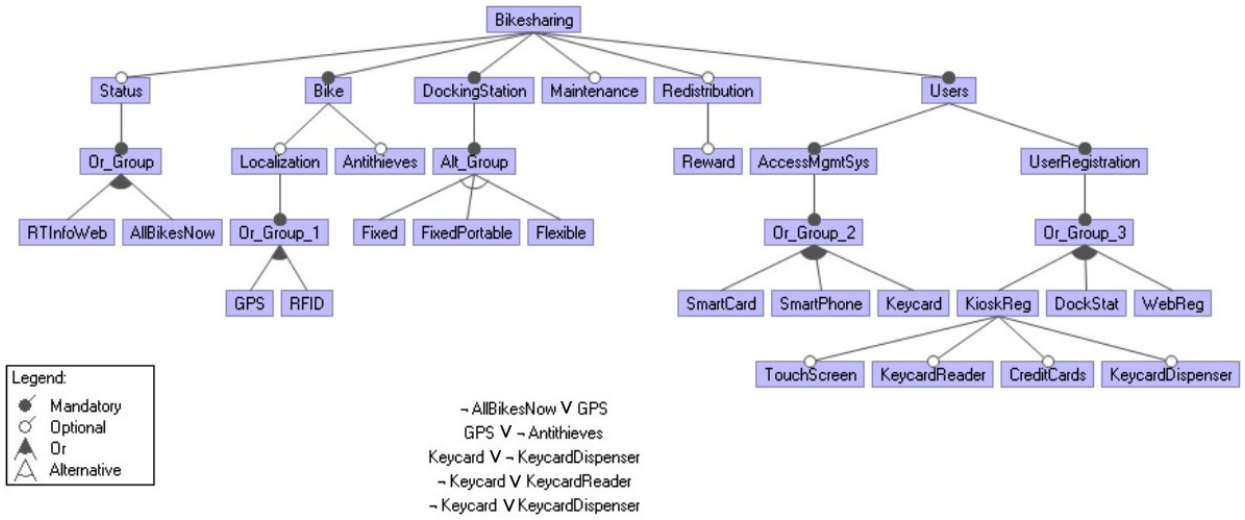

Figure 1: The complete BSS Feature Model in FeatureIDE

The deployment of such different measures gives rise to numerous multi-objective optimization problems, making this case study an interesting benchmark for QUANTICOL.

In order to apply variability analysis techniques to the case study, we develop an initial feature model by taking into account the main characteristics (features) of BSS as described in the literature (mainly [14]) and from our collaborations, in the context of QUANTICOL, with PisaMo S.p.A., an in-house public mobility company of Pisa's administration that has introduced the public bike-sharing system *CicloPi* in Pisa two years ago, and its supplier Bicincittà S.r.l. (www.bicincitta.com). These features include bikes equipped with an optional localization feature (RFID or GPS) and an optional antithieves feature (which requires GPS), docking 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 incentive scheme based on rewards. Obviously we could take many more characteristics of BSS into account, but the chosen ones represent a sufficient starting point for our exploratory study.

The resulting feature model is represented in Fig. 1, created with the FeatureIDE [19] feature model editor. From this model, all possible valid configurations, corresponding to different deployment of specific systems, may be derived.

Different configurations exhibit different values of quantitative objectives of interest to the stakeholders. In particular with respect to cost, customer satisfaction, capacity and security, sometimes the interests of users and providers of the BSS are in conflict.

3. QUANTITATIVE MODELING

Until now we only considered ordinary feature models, i.e., feature diagrams modeling 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 CAS, i.e., we consider the behavior of the components of a CAS 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 add attributes and quantitative constraints over attributes to our BSS specification and consequently perform quantitative analyses, i.e., we consider the modeling and analysis of attributed feature models.

3.1 Clafer

Clafer is a general-purpose modeling language designed to represent domains, meta-models, components and variability models [2, 3, 4]. It has successfully been used to model and optimize product lines [1, 16]. In [15], it has been used for architecturally modeling a realistic automotive scenario.

Clafer allows attributed feature modeling. 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.

3.2 ClaferMOO

The ClaferMOO extension of Clafer was specifically introduced to support attributed feature models as well as the resulting complex multi-objective optimizations [1, 15, 16]. 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 specifically designed to support product line scenarios.

3.3 Adding Attributes to the BSS

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

We thus annotate the features of the BSS (cf. Fig. 1) with attributes and to define some global quantitative constraints over these attributes. For now, we limit ourselves to the cost and customer satisfaction of features and, in specific cases, their capacity and security. Consequently, four global constraints aim to minimize the total cost of a configuration while maximize the customer satisfaction, capacity and security of a BSS. The cost, capacity and security attributes were extracted from public documents provided by PisaMo and Bicincittà concerning *CicloPi* and from documents describing the cost of similar BSS implemented in other Italian cities. The customer satisfaction attribute values were extracted from the results of an online poll among future users of a BSS in Pisa concerning which features were considered most useful to them [9, 17].

3.4 Clafer Model

The BSS is modeled in Clafer by initially categorizing each feature of the BSS into one of three basic types: `Feature`, `CapacityFeature` and `SecurityFeature`. A clafer type for each kind of feature is defined (cf. Listing 1). We use inheritance to express the fact that Security and Capacity features also have a cost and an impact on customer satisfaction.

Listing 1: Model Types

```
abstract Feature
  customersat : integer
  cost : integer

abstract CapacityFeature : Feature
  capacity : integer

abstract SecurityFeature : Feature
  security : integer
```

The full Clafer model (cf. Listing 2) contains a main abstract clafer BIKES. Each of its subfeatures is an implementation of one of the previously defined types. The attribute values correspond to a BSS with 6 stations, for a total of 60 parking lots, and 30 bikes.

Listing 2: Full Clafer Model

```
abstract BIKES
    or Status : Feature ?
        [ customersat = 0 ]
        [ cost = 0 ]
        RTInfoWeb : Feature
            [ customersat = 10 ]
            [ cost = 1000 ]
        AllBikesNow : Feature
            [ customersat = 20 ]
            [ cost = 1000 ]
    Bike : SecurityFeature
        [ customersat = 0 ]
        [ cost = 0 ]
        [ security = 0 ]
        or Localization : SecurityFeature ?
            [ customersat = 0 ]
            [ cost = 0 ]
            [ security = 0 ]
        RFID : SecurityFeature
            [ customersat = 10 ]
            [ cost = 90 ]
            [ security = 1 ]
        GPS : SecurityFeature
            [ customersat = 15 ]
            [ cost = 220 ]
            [ security = 8 ]
    Antithieves : SecurityFeature ?
        [ customersat = 5 ]
        [ cost = 90 ]
        [ security = 15 ]
    Basket : Feature ?
        [ customersat = 25 ]
        [ cost = 240 ]
xor DockingStation : CapacityFeature
        [ customersat = 0 ]
        [ cost = 0 ]
        [ capacity = 0 ]
        Fixed : CapacityFeature
            [ customersat = 17 ]
            [ cost = 48000 ]
            [ capacity = 60 ]
        FixedPortable: CapacityFeature
            [ customersat = 20 ]
            [ cost = 60000 ]
            [ capacity = 90 ]
        Flexible: CapacityFeature
            [ customersat = 23 ]
            [ cost = 90000 ]
            [ capacity = 120 ]
Maintenance : Feature ?
    [ customersat = 15 ]
    [ cost = 2000 ]
Redistribution : Feature ?
    [ customersat = 15 ]
    [ cost = 4000 ]
    Reward : Feature ?
        [ customersat = 5 ]
        [ cost = 1000 ]
Users : Feature
    [ customersat = 0 ]
    [ cost = 0 ]
    or  AccessMgmtSys : Feature
            [ customersat = 0 ]
            [ cost = 0 ]
            SmartCard : Feature
                [ customersat = 7 ]
                [ cost = 7200 ]
            SmartPhone : Feature
                [ customersat = 10 ]
                [ cost = 1140 ]
            Keycard : Feature
                [ customersat = 7 ]
                [ cost = 4860 ]
    or  UserRegistration : Feature
            [ customersat = 0 ]
            [ cost = 0 ]
            KioskReg : Feature
                [ customersat = 10 ]
                [ cost = 10 ]
                TouchScreen : Feature ?
                    [ customersat = 10 ]
                    [ cost = 480 ]
                KeycardReader : Feature ?
                    [ customersat = 5 ]
                    [ cost = 240 ]
                CreditCards : Feature ?
                    [ customersat = 13 ]
                    [ cost = 240 ]
                KeycardDispenser : Feature ?
                    [ customersat = 20 ]
                    [ cost = 600 ]
            DockStat : Feature
                [ customersat = 15 ]
                [ cost = 20 ]
            WebReg : Feature
                [ customersat = 15 ]
                [ cost = 40 ]
[ Antithieves => GPS ]
[ AllBikesNow => GPS ]
[ KeycardDispenser <=> Keycard ]
[ Keycard => KeycardReader ]
```

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

Constraints are expressed as logic propositions on the presence of features, e.g., `Antithieves => GPS` indicates that `GPS` is required whenever `Antithieves` is present. The clafers `total_customersat`, `total_cost`, etc., do not represent features but instead contain the total value of each attribute for a product instance, e.g., the total cost of the product is calculated as the sum of the cost of each feature. When ClaferMOOVisualizer instantiates all concrete products of a model, these clafers are used to compare product instances and to establish their position in the Pareto front during multi-objective optimization.

Finally, the last section of the model contains the set of optimization goals (cf. Listing 3). To optimize the model, Clafer needs an instance (`Mybike : BIKES`) and a set of goals, e.g., `<< max Mybike.total_customersat >>` to maximize customer satisfaction.

Listing 3: Optimization Goals

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

3.5 Visualization

ClaferMOOVisualizer is a web-based tool that allows users to load a Clafer Model to visualize and compare instances of the model. It comes with the following six visualizations. *Objectives and Quality Ranges* shows the optimization objectives, allowing to filter by quality. It shows, e.g., that the total cost of our BSS ranges from 49150 to 116470 euro. *Bubble Front Graph* is a bubble chart where each bubble represents an instance (also called variant). The graph supports up to four dimensions: the x- and y-axes and the color and size of the bubbles. *Feature and Quality Matrix* lists all the variants and their properties. In *Parallel Coordinates Chart*, each variant is represented by a line and each attribute is represented by a vertical axis. The point of intersection between a line and the axis corresponds to the value that the attribute holds for that variant. The latter three together allow to observe the entire Pareto front, to filter by features or quality and to select a subset of variants. Finally, *Variant Comparer* and *Spider Chart* show the commonalities and differences of two or more selected variants, allowing to perform trade-off analyses on selected variants.

The bubble chart resulting from our Clafer Model is shown in Fig. 2 and is composed of 249 variants. Customer satisfaction is shown on the x-axis and total cost on the y-axis. The size of each bubble represents the security of the variant, where bigger bubbles have higher security. The color of each bubble is linked to the capacity of the stations: since three types of stations are available (`Fixed`, `FixedPortable`, `Flexible`) each with a different capacity (10, 15 or 20 parking slots per station, respectively), this attribute is expressed by means of three colors: red for capacity 10, yellow for capacity 15 and green for capacity 20. The resulting graph contains variants disposed in three clusters.

Differences	22	1	30
Fixed ?	⊘	✓	⊘
customersat	-	17	-
cost	-	48000	-
capacity	-	60	-
FixedPortable ?	✓	⊘	⊘
customersat	20	-	-
cost	60000	-	-
capacity	90	-	-
Flexible ?	⊘	⊘	✓
customersat	-	-	23
cost	-	-	90000
capacity	-	-	120
total_customersat	252	249	255
total_cost	86470	74470	116470
total_capacity	90	60	120

Figure 3: Variant Comparer - three clusters

Differences	125	159	164
Mybike.Bike.Localization.GPS.customersat	15	15	-
Mybike.Bike.Localization.GPS.cost	2220	2220	-
Mybike.Bike.Localization.GPS.security	8	8	-
Mybike.Bike.Antithieves.customersat	5	-	-
Mybike.Bike.Antithieves.cost	90	-	-
Mybike.Bike.Antithieves.security	15	-	-
Mybike.total_customersat	112	107	92
Mybike.total_cost	51760	51670	49450
Mybike.total_security	23	8	0

Figure 4: Variant Comparer: same cluster

Using the Variant Comparer we first select an instance of each cluster in Fig. 2 and identify the different docking stations as the features differentiating the three clusters. The three variants (1, 22 and 30) are those with the highest customer satisfaction in its respective cluster. But, as we can see in Fig. 3, such levels of customer satisfaction come at a high price. Raising the total capacity is what shifts the cluster upwards, having a relatively low impact on customer satisfaction but a high impact on cost.

In Fig. 4, instead, we compare three instances of the lowest cluster with respect to `GPS` and `Antithieves`. Variant 164 contains only basic security features, variant 159 contains `GPS` and, finally, variant 125 contains both `GPS` and `Antithieves` (a variant with solely `Antithieves` does not exist, as a constraint in the model imposes the presence of `GPS` whenever `Antithieves` is selected). The presence of these security features seriously impacts the security level and has a considerable impact on the customer satisfaction as well, while the impact on the total cost of the product is much more limited.

Likewise we could compare the top-right variant of a cluster with the bottom-left variant of the same cluster, i.e., the variants with the highest and lowest cost of the respective cluster. The former variant contains all optional features and, while costing considerably more, its customer satisfaction is also roughly six times higher. Intuitively, selecting all optional features while leaving out the security-related features yields products with a better cost/satisfaction ratio.

Many more conclusions can be drawn by close inspection.

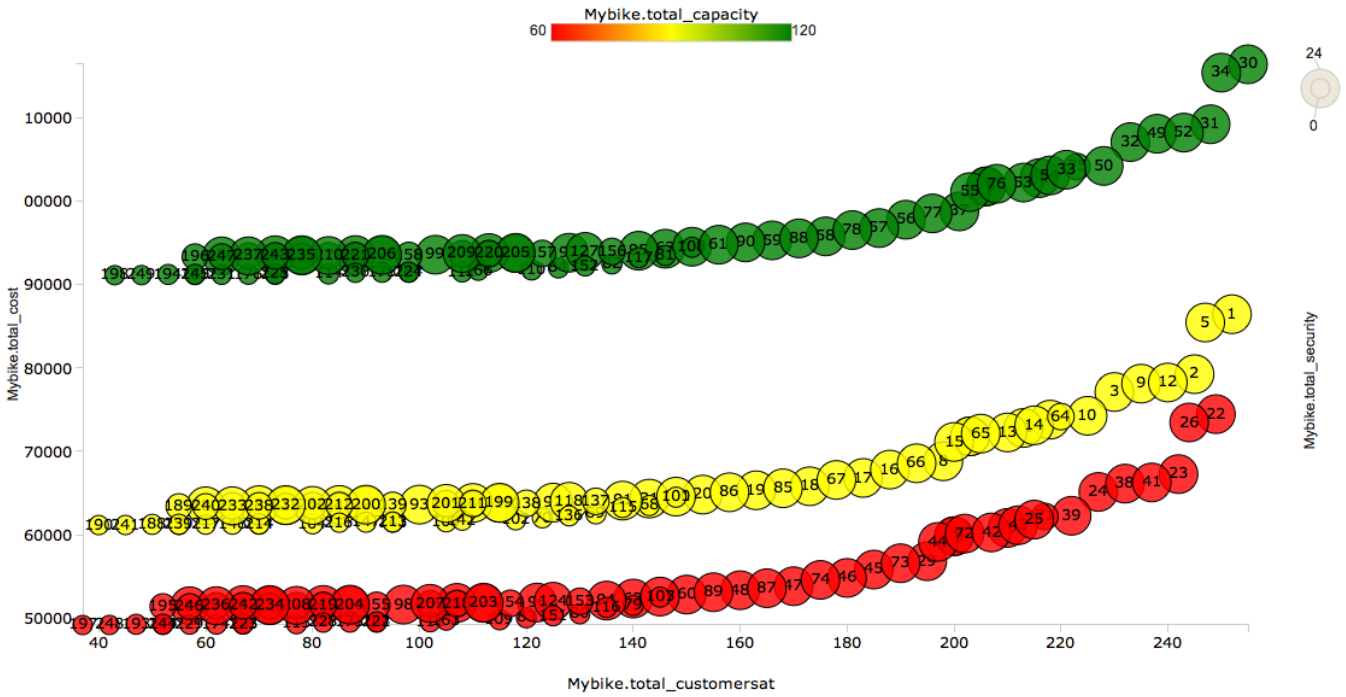

Figure 2: Bubble Front Graph of BSS Model Optimization with ClaferMOOVisualizer

4. COMPARISON WITH A CLASSICAL DESIGN APPROACH

The implementation of a BSS proposed in [17] uses an approach based on classical service design principles, composed of a sequence of steps: an initial Opportunities Identification stage, Feasibility Analysis, Conceptual Design, Detailed Design and a final stage of Service Testing and Improvement.

The Opportunities Identification stage is mostly aimed at identifying market needs and opportunities. Documents produced during this stage are the result of market research, requirements elicitation and identification of the main competitors and other forces that participate in the market.

The Feasibility Analysis aims to determine whether there are opportunities in the market that are not matched by services offered by competitors, and whether developing products to match these opportunities could be economically advantageous.

During the Conceptual Design stage, various product concepts are considered after which a single product is selected for further development. Product concepts are generated manually, reasoning on the results of earlier analyses and using brainstorming and conceptual maps. The decision making process is supported by the documents redacted during the previous stages, each one describing a different aspect of the system under development, such as a comparison with other implementations, customer satisfaction, costs, possible improvements, etc. New data might be collected if necessary.

The final stages, Detailed Design and Service Testing and Improvement, lead to the development of the first prototype followed by testing and fine-tuning. Only minor refinements to the specifications are allowed at this stage.

Variability plays an important role in the Conceptual Design stage, as each product concept is in essence a variant of the BSS. Since features and attributes are scattered among documents with a different structure, it is difficult to have a high-level view of the system being built. Also, since variability is not specifically managed, it must be inferred from the documents and variants have to be generated manually. The lack of a structured set of attributes might yield incomplete data, e.g., a feature proposed by users of an online poll might be highly desirable but, since it is not described elsewhere, there might be no information concerning its cost. In the approach proposed in [17], this problem is partially solved by refining the documents during this stage by means of new research. However, as the number of optional features increases, the documents typically grow in size and level of detail and it becomes harder to keep track of all the features and attributes. Moroever, since trade-off analyses are performed manually, without the assistance of graphical tools, some product concepts might be erroneously excluded.

By using a flexible modeling language such as Clafer to represent the complete set of selected and optional features, stakeholders are able to (i) have a more clear overview of the full system being developed, (ii) better manage variability and (iii) effectively discuss the impact of changes to the set of selected features. The graphical representation of Pareto-optimal variants (cf. § 3.5) turns out to be a valuable and user-friendly tool to manage trade-offs, especially when dealing with a high number of variants.

5. RELATED WORK

In [7], a preliminary study is described towards automatic decision support for the initial design of a BSS, as well as its successive adaptations and reconfigurations, and which takes both qualitative and performance aspects into consideration. It sketches two orthogonal approaches for the evaluation of BSS designs using automated tools. Similar to the current paper, the Clafer toolset is used for variability analysis on

BSS configurations, whereas the recently developed mean field model checker FlyFast [12] is used for performance analyses on behavioural BSS models. A common future goal is to strengthen the integration of such approaches and use the outcome of performance analyses as input for variability modeling. One could, e.g., measure the user satisfaction for specific configurations and feed the resulting values into a variability model. Performance analyses such as those performed in [7] (e.g., *the probability of finding an empty/full docking station given the specific capacity of a BSS configuration*) clearly have an impact on user satisfaction.

The choice for using the Clafer toolset also in this paper was made mainly on the basis of its ready availability and immediate usability. We found the toolset powerful enough to support the analysis of our product line case study.

For now, we did not consider the adoption of more sophisticated techniques for multi-objective optimization that have recently gained a lot of interest in the field of Search-Based Software Engineering (SBSE) applied to Software Product Line Engineering (SPLE), like genetic algorithms and multi-objective evolutionary algorithms [18]. We refer to [10, 13] for recent overviews on the interplay between the SBSE and SPLE disciplines. We noticed that most of the work has focused on computing optimal test suites in testing performed at the level of domain engineering. Hence the objectives to optimize are somewhat 'internal' to the software architecture. Returning to our choice for Clafer tool support, [10] confirms that more extensive and robust tool support for SBSE/SPLE is needed and that Clafer and ClaferMOO currently are the only tools that are readily available.

Our case is actually a system engineering product line, and the optimization objectives are defined at system level and mostly address the needs of the system's final users and stakeholders. It is left to future work to establish how beneficial (if at all) the aforementioned SBSE techniques could be to this kind of product lines.

6. CONCLUSIONS

Quantitative analysis has recently started to gain popularity for performing variability analyses in product line engineering. The experience reported in this paper, although at a preliminary stage, has shown the general value of product line modeling when addressing a class of complex systems, namely CAS, given its ability to identify commonalities and variabilities between the elements of the class. The additional possibility to base quantitative evaluation and optimization techniques on product line modeling seems to offer a promising opportunity.

7. ACKNOWLEDGMENTS

Research supported by EU project QUANTICOL (600708) and MIUR project CINA (PRIN 2010LHT4KM). We thank Bicincittà S.r.l. and Marco Bertini of PisaMo S.p.A. for sharing valuable information concerning Pisa's BSS and Simone Della Longa for his contribution to the Clafer modeling.

8. REFERENCES

[1] M. Antkiewicz, K. Bąk, A. Murashkin, R. Olaechea, J. Liang, and K. Czarnecki. Clafer tools for product line engineering. In *SPLC'13*. ACM, 2013, 130–135.

[2] K. Bąk. Modeling and analysis of software product line variability in Clafer. PhD thesis, U. Waterloo, 2013.

[3] K. Bąk, K. Czarnecki, and A. Wąsowski. Feature and meta-models in Clafer: mixed, specialized, and coupled. In *SLE'10*. *LNCS* 6563, Springer, 2010, 102–122.

[4] K. Bąk, Z. Diskin, M. Antkiewicz, K. Czarnecki, and A. Wąsowski. Clafer: unifying class and feature modeling. *Softw. Syst. Model.*, 2015.

[5] M.H. ter Beek, L. Bortolussi, V. Ciancia, S. Gnesi, J. Hillston, D. Latella, and M. Massink. A quantitative approach to the design and analysis of collective adaptive systems for smart cities. *ERCIM News* 98, 2014.

[6] M.H. ter Beek, A. Fantechi, and S. Gnesi. Challenges in modeling and analyzing quantitative aspects of bike-sharing systems. In *ISoLA'14*. *LNCS* 8802, Springer, 2014, 351–367.

[7] M.H. ter Beek, S. Gnesi, D. Latella, and M. Massink. Towards automatic decision support for bike-sharing system design. 2015.

[8] P. DeMaio. Bike-sharing: history, impacts, models of provision, and future. *J. Public Transp.* 12, 2009, 41–56.

[9] L. Gianfrotta and S. Topazzini. Progettare servizi pubblici: elaborazione di un modello per lo sviluppo di nuovi servizi e sua applicazione al caso bike sharing di Pisa. Master's thesis, U. Pisa, 2013. Italian.

[10] M. Harman, Y. Jia, J. Krinke, W.B. Langdon, J. Petke, and Y. Zhang. Search based software engineering for software product line engineering: a survey and directions for future work. In *SPLC'14*. ACM, 2014, 5–18.

[11] J. Hillston. Challenges for quantitative analysis of collective adaptive systems. In *TGC'13*. *LNCS* 8358, Springer, 2013, 14–21.

[12] D. Latella, M. Loreti, and M. Massink. On-the-fly PCTL fast mean-field model-checking for self-organising coordination. *Sci. Comput. Program.*, 2015.

[13] R.E. Lopez-Herrejon, L. Linsbauer, and A. Egyed. A systematic mapping study of search-based software engineering for software product lines. *Inf. Softw. Technol.* 61, 2015, 33–51.

[14] P. Midgley. Bicycle-sharing schemes: enhancing sustainable mobility in urban areas. Background paper CSD19/2011/BP8, Commission Sustainable Development, UN Dept. Economic and Social Affairs, 2011.

[15] A. Murashkin. Automotive electronic/electric architecture modeling, design exploration and optimization with Clafer. Master's thesis, U. Waterloo, 2014.

[16] A. Murashkin, M. Antkiewicz, D. Rayside, K. Czarnecki. Visualization and exploration of optimal variants in product line engineering. In *SPLC'13*. ACM, 2013, 111–115.

[17] C. Niccolai and E. Zanzi. Progettare i servizi: creazione di un modello di validità generale e applicazione al servizio di bike sharing a Pisa. Master's thesis, U. Pisa, 2013. Italian.

[18] G.G. Pascual, R.E. Lopez-Herrejon, M. Pinto, L. Fuentes, A. Egyed. Applying multiobjective evolutionary algorithms to dynamic software product lines for reconfiguring mobile applications. *J. Syst. Softw.* 103, 2015, 392–411.

[19] T. Thüm, C. Kästner, F. Benduhn, J. Meinicke, G. Saake, T. Leich. FeatureIDE: an extensible framework for feature-oriented software development. *Sci. Comput. Program.* 79, 2014, 70–85.