Variability-Based Design of Services for Smart Transportation Systems

Maurice H. ter Beek[1], Alessandro Fantechi[1,2], Stefania Gnesi[1], and Laura Semini[1,3]

[1] ISTI–CNR, Pisa, Italy
[2] University of Florence, Italy
[3] University of Pisa, Italy

Abstract. A smart transportation system can be seen as an aggregate of transportation opportunities and services, accompanied by advanced management services that make the access to the system easier for the user. In this paper, we exploit the product line paradigm to address the variability of an exemplary smart transportation system: a bike-sharing system. Improving the satisfaction of a user of a bike-sharing system includes providing information at runtime on the filling degree of the docking stations in the near future. To fulfill this expectation, a prediction service is needed to infer the probability that at a certain time of the day a user will return a bike to or take one from a station. In earlier studies, several possible advanced smart predictive services were identified. The choice of which services to offer to users by the managers of a bike-sharing system is influenced by minimizing the costs while maximizing customer satisfaction. To aid the managers, we modeled a family of smart bike-sharing services, after which an attributed feature model was used to augment the model with quantitative attributes related to cost and customer satisfaction, allowing for a multi-objective optimization by dedicated tools. We observe that the performance of the smart prediction services, and therefore of the related customer satisfaction, is highly dependent on the amount of collected historical data on which the predictive analysis is based. Therefore the result of the optimization also depends on this factor, which evolves over time.

1 Introduction

A smart transportation system can be considered as an aggregate of transportation opportunities and services, accompanied by advanced management services that make the access to the system easier to the final user. An example is a bike-sharing system. Bike-sharing systems are becoming more and more popular as a sustainable means of smart transportation in urban environments [19]. Ever more advanced services that aim to improve their usability are being developed and they moreover constitute a challenging case study with numerous interesting optimization problems [6, 8–14, 16, 21]. Many cities are currently adopting fully automated public bike-sharing system as a green urban mode of transportation. The concept is simple: a user arrives at a docking station, takes a bike (using e.g. an RFID card), uses it for a while and eventually returns it to a station.

Improving the satisfaction of a user of a bike-sharing system includes informing her about the status of the stations at runtime. Most current systems provide information in terms of the number of bikes parked in each station, by means of services available via the Web. In case the arrival station is full or the departure station is empty, the user might be pleased to know it if there is any chance that this situation will change within a reasonable amount of time. To fulfill this expectation, a prediction service is needed to infer the probability that at a certain time of the day a user is going to return a bike to or to free a slot of a station. Different kinds of prediction services, with different cost and performance, can be envisaged for this purpose, using e.g. machine-learning techniques.

As part of a collaboration with the company that has installed a bike-sharing system in Pisa, two studies were conducted recently, with the aim of applying advanced design techniques to provide the users with a better service. In the first study [7], the product line paradigm was applied, at the level of systems engineering, in order to jointly address variability issues and quantitative analysis of the possible options that a generic bike-sharing system can exhibit. To this aim, first a reference feature model was defined and then feature attributes and global quantitative constraints were added, thus creating an attributed feature model suitable to conduct multi-objective optimization analyses. The focus was on the selection of physical components of the overall system, like the number and kind of stations. In the second study [3, 4], a series of advanced predictive software services for the user of a bike-sharing system were envisaged, and partly implemented, using a machine-learning approach to analyze usage patterns and to learn computational models of such services from logs of actual system usage.

Here we aim to combine these efforts by proposing a product line framework defining a family of advanced prediction services. The services are evaluated with respect to their performance, expressed in terms of the accuracy of the prediction [3, 4], on which customer satisfaction depends. The accuracy is the outcome of the use of several machine-learning techniques, based on the availability of data logs from system usage. The multi-objective analysis as applied in [7] is then used to assist the maintainers of the system in choosing the optimal configuration.

It turns out that the performance of the prediction services, and therefore of the related customer satisfaction, is highly dependent on the amount of collected historical data on which the predictive analysis is based. Therefore the result of the optimization also depends on this factor, which evolves over time. A typical scenario is the one of a newly installed bike-sharing system that initially has no historical data, and hence predictive services are inaccurate (and therefore have a low benefit/cost ratio) and only after years of usage the collected historical data can make a predictive service accurate enough to be safely provided to the user. The quantity of data needed may differ from service to service, though. Hence, the optimal configurations of the product line of services for bike-sharing systems evolves over time as well. The idea is to gradually introduce more advanced prediction services as over time more data on the system in operation becomes available. The availability of a set of tools to support this evaluation will aid the maintainers of the bike-sharing system in dynamically choosing the optimal configuration during its life cycle.

Based on concrete experiments with data concerning the bike-sharing system of Pisa (our 'testbed'), we show how preliminary results teach us that the relation between the size of available logged usage data and performance is not easy to capture with machine-learning techniques. Actually, it turned out that below a given threshold (a few months of usage) the prediction service is highly unreliable. We therefore tailored the analysis techniques applied in [7] to cope with this discrete relation between the size of available data and performance indicators. The described techniques are actually considered as an initial step in a research activity aimed at a full study of these kind of dependencies. Nevertheless, in our opinion, the resulting variability-based design process can be applied not only to bike-sharing systems of other cities, but in general to smart transportation systems which rely on the availability of advanced prediction services.

The paper is organized as follows. In Sect. 2, we introduce the bike-sharing system testbed. In Sect. 3, we discuss possible new smart prediction services that could be added to the bike-sharing system and we show how to estimate their performance based on machine-learning techniques. In Sect. 4, we address the quantitative analysis and optimization issues by adding quantitative attributes to features (e.g. performance in case of the prediction services) and show how to perform multi-objective quantitative analyses of the system. In Sections 5 and 6, we discuss the results of our preliminary analyses and conclude the paper.

2 A Smart Transportation System

In this section, we briefly describe Pisa's bike-sharing system *CicloPi*. It will be the running example in this paper. *CicloPi* was introduced in town three years ago by PisaMo S.p.A., an in-house public mobility company of the Municipality of Pisa, with whom we started a fruitful collaboration. This bike-sharing system, which controls some 150 bikes and 15 stations, was supplied by Bicincittà S.r.l., together with a few basic services, described next. This is hence a relatively small bike-sharing system, which makes it an excellent testbed for our research.

Currently, *CicloPi* offers two basic services, one reserved to its administrators and one meant for its users. The first, which we call `Bikes Hired`, registers, for each hired bike, the user ID and the departure station and time of departure. It is needed to control whether all bikes are returned and it makes it possible to offer a billing service to the administrators (e.g. a fee applies for rides longer than 30 minutes). Up to now this service has collected way more than 300.000 entries of the following form: $\langle$ `ID`, $\underbrace{\texttt{station}, \texttt{slot}, \texttt{date}\,\&\,\texttt{time}}_{departure}$, $\underbrace{\texttt{station}, \texttt{slot}, \texttt{date}\,\&\,\texttt{time}}_{arrival}$ $\rangle$

In [3, 4], we exploited about 280.000 entries of this form, collected in two years, to design a number of predictive services, discussed in the next section.

The second basic service of *CicloPi* that is currently available is one that we call `Stations Snapshot`. It allows the user to perform a real-time check, for each docking station, of the availability of bikes at that station. Users can access this service using a Web browser to find the closest station with an available bike (or with a free slot, for returning a bike) before actually going there.

We note that the bike-sharing system administrators can also use this service, e.g. to plan redistributions. Redistribution of bikes (usually by trucks) is used to balance the number of bikes: bikes are taken from (nearly) saturated stations and parked in (nearly) empty stations. As such, the usual flow of bikes from one group of stations to another (e.g., from stations located in residential areas to those located in work areas in the morning and vice versa in the afternoon) can be balanced. However, to do this efficiently is a major issue for bike-sharing systems, but out of the scope of this paper (cf. [8, 12, 14]).

3 New Smart Prediction Services

To improve the satisfaction of users of a bike-sharing system, we propose several new smart services that provide information at runtime on the filling degree of the docking stations. In our testbed, customer satisfaction may be enhanced by offering hints on the status of the stations in the next few minutes. If the station of interest is currently empty, a user may be pleased to know that, with a high probability, one of the circulating bikes will be returned there soon. Likewise, for stations that are full it may be worthwhile to know the chances that someone will soon free a slot by renting a bike. However, vendors and installers of such bike-sharing systems would likely prefer to analyze and assess different solutions before actually putting them into operation in a specific setting.

To this aim, we propose to organize the new services as a family of services in an *attributed feature model* (i.e. a feature model enriched with attributes containing non-functional information on features). Feature models are a useful means to structure systems with variability. In our case study, the attributed feature model supports the stakeholders in understanding the system architecture and in deciding which product of a line to deploy, since we provide an estimate of the cost and customer satisfaction for each variant. We base the customer satisfaction attribute on a predictive performance measure, which is obtained by applying machine learning to analyze usage patterns and learn computational models of the respective features from logs of usage of the bike-sharing system.

Machine learning (ML) is concerned with computational models and methodologies to realize data-driven adaptive approaches to data analysis, pattern discovery and recognition, as well as to the predictive modeling of input-output data relationships. The term data-driven refers to the fact that ML approaches rely on (numerical) information encoded in the data [15]. In our case, ML methodologies allow us to learn the (unknown) relationship between the feature and its inputs by exploiting historical data representing examples of such input-output map. A trained model can then be used to provide predictions on future values of the feature in response to new input information, i.e. providing an implementation of the feature. The advantage of such an approach is twofold. On the one hand, such methodologies provide a powerful means to realize a wide choice of predictive features for which there exists meaningful historical data. On the other hand, trained ML models are provided with a measure of predictive performance that can be used as a metric to assess the customer satisfaction of the feature.

In [3, 4], we put forward the idea of using ML methodologies to implement a preview service for a bike-sharing system. Here we combine this solution with the design capabilities offered by software product line engineering, exploiting the second characteristic of an ML approach, viz. the ability of trained ML models to return a performance measure.

3.1 A Feature Model of a Family of Services

Our starting point is the current set of basic services offered by *CicloPi*, viz. `Bikes Hired` and `Stations Snapshot` concerning the status of bikes and stations.

Subsequently, we add several further preview services, described next, which can be offered in the presence of a minimum of historical data.

The feature model of the resulting family of services is depicted in Fig. 1. The existing services that keep the status of the bike-sharing system updated are mandatory, while the new smart prediction services are optional.

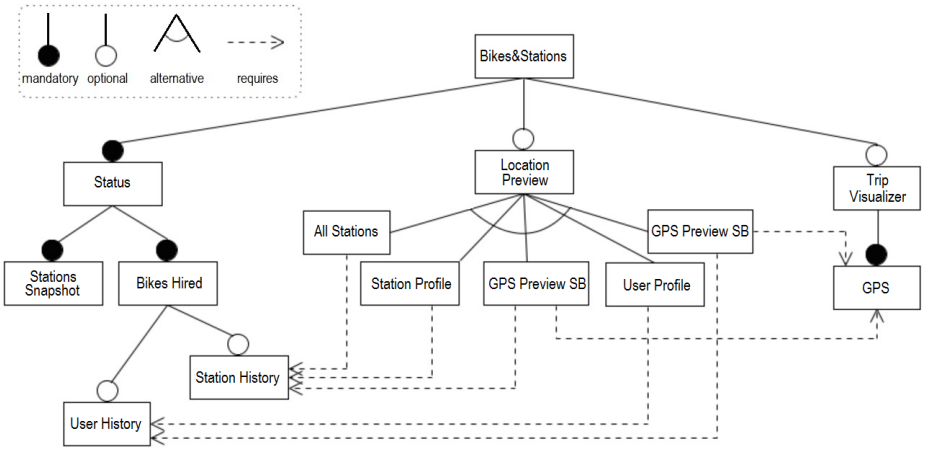


Fig. 1. Feature model for a product line of services of a bike-sharing system

The `Location Preview` service is concerned with the prediction of the destination of a circulating bike and is implemented using one of its sub-features, all of whose services are based on training ML models. They differ in the data they use and consequently in their performance, as we discuss in this section.

The services `All Stations`, `Station Profile` and `User Profile` use historical data, but they differ in the aggregation/disaggregation of data: the first being the simplest to implement and maintain but with the lowest predictive performance. They require one of the `Station History` and `User History` features of the basic `Bikes Hired` service, based on the type of profile.

The station-based and user-based services `GPS Preview SB` and `GPS Preview UB`, respectively, combine historical data with traces of the circulating bikes as communicated by GPS trackers, which is why they require `GPS` (trackers) next to the `Station History` and `User History` features of the basic `Bikes Hired` service, based on the type of profile. We will describe these services (sub-features of `Location Preview`) in Sect. 3.2, after an introduction to relevant ML techniques.

`Trip Visualizer`, finally, is a service that permits to track the circulating bikes, and to offer a trace visualization service to the user so that she can have the real-time position and direction of all bikes and figure out if one is approaching. Obviously, to implement this feature, each bike must have a GPS tracker.

GPS trackers are already in use in some bike-sharing systems, either as anti-theft or to eliminate docking stations: bikes are parked and locked anywhere, a user locates the closest one using the GPS and unlocks it with a code.

3.2 Machine Learning for Location Preview

ML is an active and wide research field comprising several paradigms, e.g. neural-inspired, probabilistic, kernel-based approaches, and addressing a variety of computational learning task types [15]. To implement the preview features we have focused in [3, 4] on ML models and algorithms targeted at solving *supervised learning tasks*. Supervised learning refers to a specific class of ML problems that comprise learning of an (unknown) map $M : \mathcal{X} \rightarrow \mathcal{Y}$ between input information $x \in \mathcal{X}$ (e.g. a vector of attributes) and an output prediction $y \in \mathcal{Y}$. Such an unknown map is learned from couples $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ of input-output data, referred to as *training examples*, following a numerical routine targeted at the optimization of a performance function which measures the quality of the predictions generated by the ML model.

ML models are characterized by two operational phases. In the *training* (or *learning*) phase, ground-truth teaching information (encoded in training samples) is used to adapt the parameters regulating the response of the model so that its performance is increased. The *testing* (or *prediction*) phase, instead, supplies a trained model with novel input information (typically unseen at training time) to generate runtime predictions. The two phases are not always disjoint: *incremental learning* approaches exist that allow to continuously adapt the parameters of a ML model, while this keeps providing its predictions in response to new input data. In general, the resulting quality of the ML model predictions is influenced, on the one hand, by the quality of the training data, which should represent a sufficient and significative sample of the relationship to be modeled, and, on the other hand, by the adequacy of the learning model for the specific learning task. In this sense, different tasks, associated with different features to be modeled, may require to use learning models with different capabilities.

The actual form of the performance measure depends on the nature of the learning task, but it typically evaluates the discrepancy between the output predicted and the actual output. Here, the most significative performance measure, which we will use in Sect. 4 to feed an attributed feature model of the feature model in Fig. 1, is an aggregated performance score (in %), called $F1$-score, calculated as follows: $F1 = \frac{2\,TP}{2\,TP+FP+FN}$, where TP, FP and FN are the number of true positive, false positive, and false negative classifications, respectively. Higher $F1$ values denote better classification performances. Other well-known performance measures are precision (the number of true positives per predicted positive) and recall (the number of true positives per real positive), of which $F1$ is the harmonic mean.

All services implicitly use the data offered by `Stations Snapshot` to combine data on actual occupancy with the prediction of the destination of circulating bikes, and have a prediction of the number of bikes that will be at a station.

ML for Location Preview: Static Data. As said before, `All Stations`, `Station Profile` and `User Profile` offer similar services. They all use data of a static type. This means that each training sample is a pair of identically and independently distributed vectors: departure and arrival data. Differences among the features depend on aggregation and/or abstraction of these data. To implement `All Stations` one learning model was built, and usage data was not partitioned into any subset, whereas to implement `Station Profile` usage data was split in order to build a different learning model for each station. In both cases the learning task abstracts from the user ID. On the contrary, `User Profile` analyzes the behaviour of single users, by building a learning model for each of them.[4]

Both `All Stations` and `Station Profile` require `Station History` to train the models (this feature maintains the history of all station-to-station movements with date and time) and the (mandatory) `Bikes Hired` feature in the prediction phase, to know the departure station of the circulating bikes.

`User Profile` uses the data maintained by `User History`, the history of all movements of each user, in the training phase, and uses `Bikes Hired` in the prediction phase, as above.

We trained and validated learning models for these services, using real-world usage data comprising more than 280.000 entries of the form discussed in Sect. 2.

The obtained results are reported and discussed in detail in [3,4]. For the purposes of the present paper, the table on the right summarizes the obtained performance measures.

Service	$F1$-score
`All Stations`	18%
`Station Profile`	34.9%
`User Profile`	69%

ML for Location Preview: Sequential Data. The `GPS Preview` features (`SB`, station-based, and `UB`, user-based) can predict the same output as the other location features considered so far, using additional input data, that is GPS trajectories corresponding to journeys performed by users of the bike-sharing system. Trajectory data encodes a form of dynamical information of different nature with respect to the static vectorial data used in Sect. 3.2, requiring a radically different ML approach. A GPS trajectory is a form of sequential data, a type of structured information where the observation at a given point of the sequence is dependent on the context provided by the preceding or succeeding elements of the sequence. Such contextual information plays a role also in the learning task where, for instance, the decision on which will be the arrival station corresponding to a GPS trajectory cannot be taken based on the observation of a single element of the sequence, but should rather take into account the context provided by the full sequence or by at least a part of it.

[4] To guarantee privacy protection, we only used anonymized data. In fact, the data from Bicincittà did not contain any information from which one could identify users.

We did not implement these features so far, due to the lack of data. Neither *CicloPi*, nor any of the bike-sharing systems provided by Bicincittà, have GPS trackers installed on the bikes. Recently, we contacted TMR S.r.l., which supplies the bike-sharing system of the municipality of Palermo. They did install GPS trackers on the bikes, to be used mainly as an anti-theft feature: bikes periodically communicate their position through a cellular network using the GPRS protocol. The service is brand new and they do not have a meaningful log of traces yet. As soon as this data will be available, we could decide to build learning models along the outline in [4]. We expect to obtain the scores given in the table below.

Service	$F1$-score
`GPS Preview SB`	41.88%
`GPS Preview UB`	82.8%

These values are based on discussions with domain experts, who foresee that the performance of the learning models will be improved by roughly 20% with respect to the corresponding models built so far based on static data alone.

In the remainder of this section, we briefly describe how we would go about implementing the use of sequential data. A straightforward approach is to use models for static data feeding them with a fixed-size chunk of the input sequence. This window of observations can be slid across the full length of the sequence, providing a prediction for each sequence element that can take into consideration the surrounding elements up to the window length. The key issue of the approach is how to determine the correct size of the window for each learning problem. To address this issue, several learning models have been proposed that are capable of maintaining a memory of the history of the input signals and to use it to compute their predictions. We now describe one such model.

Recurrent Neural Networks (RNN) [17] were proposed specifically to deal with the dynamics of sequential information. They extend the original artificial neural networks paradigm with feedback connections that introduce a dynamic memory of the neuron activation which can be used to encode short to long term dependencies among the elements of the sequence, depending on the specific network architecture. In this context, the use of Reservoir Computing (RC) [18] gained increasing interest, due to its ability in conjugating computational efficiency with the RNN capability of dealing with learning in temporal sequence domains. The underlying idea of the RC approach is to use a layer of sparsely connected recurrent neuron whose connections are initialized and left untrained; adaptation of the neural weights is restricted to the layer of output neurons. This allows to considerably reduce the computational complexity of training, which is a key issue if performed at runtime. RC models appear well suited for the implementation of the `LocationPreview` feature. In particular, they already showed considerable efficacy in closely related learning tasks, like the prediction of the destination room of trajectories of users walking in indoor environments [2].

4 A Family of Smart Services

In this paper, we set out to combine and extend the ideas of [3,4,7] by proposing a product line of a family of advanced prediction services. The services can be distinguished by the accuracy of the prediction obtained by using different ML

techniques. To perform quantitative analyses, we add non-functional attributes and quantitative constraints over attributes to the feature model of our bike-sharing system (thus turning it into an attributed feature model).

4.1 Clafer and Multi-Objective Optimization

Clafer is a general-purpose modeling language designed to represent domains, meta-models, components and variability models [5], including attributed feature models [1]. It can be used to model and optimize product lines [1,20]. Clafer supports partial instances, i.e. a user can define a partial model with some undecided variability. The complete set of instances can be generated from the partial model. Alternatively, a user can define concrete instances with no variability.

A Clafer model is composed of definitions called *clafers*, which can represent either properties, types or references, depending on their nesting and syntactic modifiers. Here we use them to model features and from now on we speak of features rather than clafers. Constraints express dependencies among features or restrict allowed values. A feature can contain one or more sub-features and can include multiplicity, i.e. the number of times that the feature can be instantiated (not considered here). Optionality is denoted using a question mark. An optional feature may or may not be present in model instances.

Each feature can have one or more associated attributes and quality constraints can be specified either globally or in 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. Imagine, e.g., that each feature also has a value for user satisfaction 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 and in particular the resulting complex multi-objective optimization goals [1, 20]. A multi-objective optimization problem has a set of solutions, known as the Pareto front, representing 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. 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 [1,20].

4.2 Clafer Model of a Family of Services

We specified the attributed feature model of Fig. 1 in Clafer. Each feature of the bike-sharing system is defined as a clafer of the basic type `Feature` with three attributes of type integer: `csat` models the customer satisfaction, `cost` models the (estimated) additional cost in euros to implement the (prediction) service and `time` models the number of months of operation of the bike-sharing system that is needed before it can become operational.

For the new prediction services, customer satisfaction reflects their performance measure, expressed in terms of the accuracy of the prediction, i.e. the $F1$-score (cf. Sect. 3), since we believe this to constitute also a good indication for customer satisfaction. The more precise the prediction, the happier the user. For this reason, we simply take the $F1$-score values (in %) returned by the experiments as the integer value of the customer satisfaction of these services, and indeed we use the same 0–100 range to indicate the customer satisfaction for the other services, i.e. the mandatory ones and `Trip Visualizer`. The total customer satisfaction is then considered as the sum of the values of the features involved. The values for these other services were extracted from the results of an online poll concerning the features considered most useful for Pisa's planned (at the time) bike-sharing system [16,21] (these features include most of the services considered in this paper) and from discussions with PisaMo and Bicincittà.

The cost attribute has a component that was obtained from the ML experiments (described in detail in [3, 4] and discussed in the previous section) and another component that was extracted from documents provided by Bicincittà and from documents concerning the cost of similar systems implemented in other Italian cities (and in one case from a private communication). The latter component concerns the additional cost required to implement the specific service.

The time attribute, finally, was derived from the ML experiments described in detail in [3, 4] and discussed in the previous section.

The full Clafer model in Listing 1.1 contains a main abstract clafer `BikesAndStations`, composed of a set of features of type `Feature`.

Listing 1.1. Full Clafer Model of *CicloPi*

```
abstract Feature
  csat : integer
  cost : integer
  time : integer

abstract BikesAndStations
  Status
    StationsSnapshot
    BikesHired
      StationHistory : Feature ?
        [ csat = 68 ]
        [ cost = 500 ]
        [ time = 0 ]
      UserHistory : Feature ?
        [ csat = 79 ]
        [ cost = 500 ]
        [ time = 0 ]
  xor LocationPreview ?
    AllStations : Feature
      [ csat = 18 ]
      [ cost = 300 ]
      [ time = 24 ]
    StationProfile : Feature
      [ csat = 35 ]
      [ cost = 400 ]
      [ time = 24 ]
    UserProfile : Feature
      [ csat = 69 ]
      [ cost = 500 ]
      [ time = 24 ]
    GPSPreviewSB : Feature
      [ csat = 42 ]
      [ cost = 100 ]
      [ time = 12 ]
    GPSPreviewUB : Feature
      [ csat = 83 ]
      [ cost = 100 ]
      [ time = 12 ]
  TripVisualizer : Feature ?
    [ csat = 35]
    [ cost = 0 ]
    [ time = 0 ]
    GPS : Feature
      [ csat = 0 ]
      [ cost = 3000 ]
      [ time = 0 ]
  [ AllStations => StationHistory ]
  [ StationProfile => StationHistory ]
  [ UserProfile => UserHistory ]
  [ GPSPreviewSB => StationHistory ]
  [ GPSPreviewSB => GPS ]
  [ GPSPreviewUB => UserHistory ]
  [ GPSPreviewUB => GPS ]

total_csat : integer = sum Feature.csat
total_cost : integer = sum Feature.cost
total_time : integer = sum Feature.time

Services : BikesAndStations

<< max Services.total_csat >>
<< min Services.total_cost >>
<< min Services.total_time >>
```

In Clafer, sub-features implicitly form an `and`-group, but they can also explicitly be declared as `or`-group or `xor`-group (i.e. modeling alternative features). For instance, `xor LocationPreview` means that any product must contain exactly one of the new smart prediction services. The additional (cross-tree) constraints are expressed as implications on the presence of features (keywords `=>` and `<=>`), e.g., `GPSPreviewSB => GPS` indicates that a product (i.e. a configuration) must require feature `GPS` whenever feature `GPSPreviewSB` is part of the product.

The clafers `total_csat`, `total_cost` and `total_time` do not represent features, but 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 of its features). When ClaferMooVisualizer instantiates all concrete products of a model, these clafers are used to compare product configurations and to place them on the Pareto front during the multi-objective optimization. To this aim, an additional part of any Clafer model contains the optimization goals. To actually optimize the model, Clafer needs a specific instance (`Services : BikesAndStations`) together with a set of goals, each of them enclosed in double brackets, e.g., `<< min Services.total_cost >>` represents the goal of `min`imizing the cost.

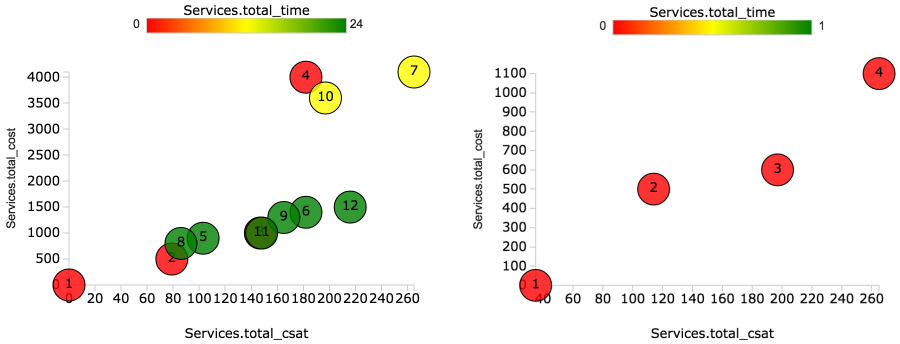
For our testbed, we are thus interested in maximizing the customer satisfaction while minimizing both the cost and the time required for the introduction of new prediction services (to become operational).

4.3 Quantitative Analysis with ClaferMoo

Now that we have an attributed feature model in Clafer, its associated tool 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. ClaferMooVisualizer is a Web-based tool that allows users to load a Clafer model and to visualize and compare instances of the model. It comes with the following four possible visualizations. *Bubble Front Graph* produces a bubble chart where each bubble represents an instance (also called variant). The graph supports up to four dimensions: the x- and y-axes, bubble color and bubble size. *Feature and Quality Matrix* lists all variants and their properties. *Variant Comparer* shows the commonalities and differences of two or more selected variants. 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. Especially the first two are very useful when one is interested in comparing a number of variants and we will illustrate their usage shortly. The third is particularly useful when one wants to understand better the subtle differences between a small number of variants, say up to three.

Evaluation of adding smart prediction services to CicloPi. The bubble chart resulting from a multi-objective optimization with ClaferMooVisualizer of the Clafer model presented in the previous section is shown in Fig. 2a.

The bubble chart is composed of 12 variants. Customer satisfaction is shown on the x-axis and cost on the y-axis, while the color of each bubble shows the time needed before the new smart prediction service of the respective variant can become operational (only assuming values 0, 12 or 24 in this case).



(a) 12 variants: bubbles 1, 2 and 4 are red (indicating `total_time` = 0); bubbles 7 and 10 are yellow (`total_time` = 12); all further bubbles are green (`total_time` = 24).

(b) 4 variants: four bubbles of the same color (red) indicating `total_time` = 0.

Fig. 2. Bubble Front Graphs of the two evaluations of adding smart prediction services

Next we inspected each of these variants by making use of the Feature and Quality Matrix reported in Fig. 3. As expected, the three variants that contain the smart `Trip Visualizer` service (4, 7 and 10) are far more expensive than the other ones, due to the need to equip all bikes with expensive GPS trackers. Variant 7, which also includes the `GPS Preview UB` service and both `Station History` and `User History`, is the variant with the highest customer satisfaction (265), which thus comes at a price. If the budget is limited, variant 12 might offer an excellent alternative with the `User Profile` prediction service, without having to give up on the either `Station History` or `User History`. It scores the second-best customer satisfaction at a reasonable cost, but the time needed to be fully operational is 24 months. This is unavoidable, though, for all new prediction services based on ML models that do not require GPS trackers.

Evaluation of adding smart prediction services elsewhere. Now suppose we want to evaluate the installation of one of the smart prediction services in a city with a bike-sharing system that has been operating for over two years and where all bikes are equipped with GPS trackers. Also in this case our setup can be useful. We set `cost(GPS)` = 0 and `time(Preview)` = 0 for all `Preview` services of `Location Preview`. The result of multi-objective optimization (effectively only two objectives in this case) with ClaferMooVisualizer is depicted in Fig. 2b.

Now all four variants obviously include the suddenly very convenient `Trip Visualizer` service (it comes for free and it is immediately operational), variants 3 and 4 also include the `GPS Preview UB` and `User History` services, whereas only variant 4 includes `Station History` (expensive, yet with a high customer satisfaction). Hence in this case, variant 3 might be the most reasonable one.

Fig. 3. Feature and Quality Matrix of the 12 variants depicted in Fig. 2a

Many more trade-offs can of course be studied in this way. In the next section, we discuss the implications of our experiments.

5 Lessons Learned and Discussion

The analysis with ClaferMooVisualizer shown in the previous section aims to provide a means to assist the management of a bike-sharing service. It offers support for the decisions as to which services are the most valuable ones to be

suggested to and eventually implemented for clients, based on a trade-off between cost and customer satisfaction mainly. However, the conducted analysis shows how a third dimension was added, viz. the deployment time which influences the availability of logged data. This is a peculiar aspect of 'smart transportation' services, which need to anticipate to the user the actual availability of a transport service, in an as accurate as possible way, in order to be accepted by them as an important added value in terms of reliability of the whole service. Predictive services based on ML clearly show that the accuracy of their forecasts are heavily dependent on the significance and size of the training set. In the smart transportation domain, these two factors depend on the amount and quality of logged data, and on the implicit assumption that the average origin/destination demand of transportation services is quite stable in the long run.

The experiments with data concerning our testbed bike-sharing system show how the relation between the size of available logged usage data and performance is not easy to capture. In fact, it turned out that below a given threshold (a few months of service, and hence of accumulated data) the prediction service is highly unreliable. We therefore tailored the analysis techniques applied in [7] to cope with three discrete steps of observation length, viz. at time zero and after one or two years. We need more experiments to elaborate a more accurate definition of the dependance of the performance indicators on the size of available data. However, the preliminary results already provide support to decide whether and when to introduce an advanced smart transportation service (in the form of a predictive service for the users) in an already deployed bike-sharing system. In fact, we considered a family of services and we indicated tools to support such decisions, corresponding to selecting the most convenient services to be offered at a given deployment time. These decisions can be reconsidered by repeating the analysis once more usage data has been accumulated.

Notice that the appeal of ML-based solutions that only use historical data is outperformed as soon as a more advanced localization technology (GPS, in this case) is able to provide more accurate real-time data. The cost of such outperforming technology is a drawback that allows ML-based surrogates to still be a convenient solution; but if the cost of the technology is absorbed by other improvements (in our case, the desire to provide anti-theft capabilities), ML-based solutions that do not exploit the real-time trajectories loose their appeal. GPS-based solutions, moreover, can be applied sooner (i.e. with less data available). For instance, `TripVisualizer` can be made available as soon as a new bike-sharing system is put in place. Solutions that apply ML techniques to analyse GPS trajectories can show a good performance also with a smaller dataset than those only using static data. Moreover, we have seen that the situation can be different for settings in which one of the (optional) features has already been foreseen (in which case it is as if it were a mandatory feature), drastically changing the outcome of the multi-objective optimization.

Note that once a more advanced predictive service should become available, then it could be included in the service family model as a new feature of the feature model, and associated with cost and customer satisfaction measures,

possibly dependent on available logged usage data. Also note that training this new service might require more detailed data, not previously accommodated for, and which hence requires to start a new data logging period. Hence, the new service could be delivered to the users only when it is mature enough according to the time-dependent performance indicators. So this is actually an evolution of the product line, followed by the evolution of the decision support analysis.

6 Conclusions

The added value of *smart transportation systems* consists in offering the user a set of services that allow for a more comfortable usage of the provided transportation means, using the most advanced technologies available. A proper trade-off between the cost of offering such services and the related customer satisfaction has to be defined to obtain the most advantage from their introduction.

Organizing the possible services in a product line offers a means to support the determination of such a trade-off. This requires quantitative analysis of the involved parameters. We moreover extended to software services a previously defined process, which was originally employed to study—by means of a quantitative analysis and at the level of systems engineering—the possible options that a generic bike-sharing system can exhibit at the physical level.

Although the experiments presented in this paper are quite preliminary, and limited to a bike-sharing system, we believe that the envisaged variability-based decision support process and tools can be adopted not only in bike-sharing systems of other cities, but in general in a wide range of smart transportation systems which rely on the availability of advanced prediction services.

7 Acknowledgments

Research supported by EU project QUANTICOL, 600708. We are very grateful to Marco Bertini of PisaMo S.p.A. and Bicincittà S.r.l. for generously sharing with us information and actual usage logs of Pisa's bike-sharing system *CicloPi*. We thank Davide Bacciu and Antonio Carta for discussions and for having conducted the machine-learning experiments.

References

1. M. Antkiewicz, K. Bąk, A. Murashkin, R. Olaechea, J. H. Liang, and K. Czarnecki. Clafer Tools for Product Line Engineering. In T. Kishi, S. Jarzabek, and S. Gnesi, editors, *SPLC*, volume 2, pages 130–135. ACM, 2013.
2. D. Bacciu, P. Barsocchi, S. Chessa, C. Gallicchio, and A. Micheli. An experimental characterization of reservoir computing in ambient assisted living applications. *Neural Comput. Appl.*, 24(6):1451–1464, 2014.
3. D. Bacciu, A. Carta, S. Gnesi, and L. Semini. Using a Machine Learning Approach in the Design of Smart Transportation Systems. *Submitted*, 2016.

4. D. Bacciu, S. Gnesi, and L. Semini. Using a Machine Learning Approach to Implement and Evaluate Product Line Features. In M. H. ter Beek and A. Lluch-Lafuente, editors, *WWV*, volume 188 of *EPTCS*, pages 75–83, 2015.
5. K. Bąk, Z. Diskin, M. Antkiewicz, K. Czarnecki, and A. Wąsowski. Clafer: unifying class and feature modeling. *Softw. Syst. Model.*, 2015.
6. M. H. ter Beek, A. Fantechi, and S. Gnesi. Challenges in Modelling and Analyzing Quantitative Aspects of Bike-Sharing Systems. In T. Margaria and B. Steffen, editors, *ISoLA*, volume 8802 of *LNCS*, pages 351–367. Springer, 2014.
7. M. H. ter Beek, A. Fantechi, and S. Gnesi. Applying the Product Lines Paradigm to the Quantitative Analysis of Collective Adaptive Systems. In D. C. Schmidt, editor, *SPLC*, pages 321–326. ACM, 2015.
8. M. H. ter Beek, S. Gnesi, D. Latella, and M. Massink. Towards Automatic Decision Support for Bike-Sharing System Design. In D. Bianculli, R. Calinescu, and B. Rumpe, editors, *SEFM*, volume 9509 of *LNCS*, pages 266–280. Springer, 2015.
9. M. H. ter Beek, A. Legay, A. Lluch Lafuente, and A. Vandin. Quantitative Analysis of Probabilistic Models of Software Product Lines with Statistical Model Checking. In J. M. Atlee and S. Gnesi, editors, *FMSPLE*, volume 182 of *EPTCS*, pages 56–70. EPTCS, 2015.
10. M. H. ter Beek, A. Legay, A. Lluch Lafuente, and A. Vandin. Statistical Model Checking for Product Lines. In T. Margaria and B. Steffen, editors, *ISoLA*, LNCS. Springer, 2016.
11. V. Ciancia, D. Latella, M. Massink, and R. Pakauskas. Exploring Spatio-temporal Properties of Bike-Sharing Systems. In *SCOPES*, pages 74–79. IEEE, 2015.
12. C. Fricker and N. Gast. Incentives and redistribution in homogeneous bike-sharing systems with stations of finite capacity. *EURO J. Transp. Logistics*, pages 1–31, 2014.
13. J. Froehlich, J. Neumann, and N. Oliver. Sensing and Predicting the Pulse of the City through Shared Bicycling. In *IJCAI*, pages 1420–1426, 2009.
14. N. Gast, G. Massonnet, D. Reijsbergen, and M. Tribastone. Probabilistic Forecasts of Bike-Sharing Systems for Journey Planning. In *CIKM*, pages 703–712. ACM, 2015.
15. L. Getoor and B. Taskar, editors. *Introduction to Statistical Relational Learning*. The MIT Press, 2007.
16. 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, Università di Pisa, 2013. In Italian.
17. J. F. Kolen and S. C. Kremer, editors. *A Field Guide to Dynamical Recurrent Networks*. IEEE Press, 2001.
18. M. Lukoševičius and H. Jaeger. Reservoir Computing Approaches to Recurrent Neural Network Training. *Comput. Sci. Rev.*, 3(3):127–149, 2009.
19. 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.
20. A. Murashkin, M. Antkiewicz, D. Rayside, and K. Czarnecki. Visualization and exploration of optimal variants in product line engineering. In T. Kishi, S. Jarzabek, and S. Gnesi, editors, *SPLC*, pages 111–115. ACM, 2013.
21. 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, Università di Pisa, 2013. In Italian.