Analyzing the Performance of Bike-Sharing Systems

Maurice ter Beek (Formal Methods and Tools, ISTI–CNR, Pisa)

Joint work with Stefania Gnesi, Mieke Massink, Diego Latella (FMT, ISTI–CNR), Axel Legay (Inria Rennes, France), Alberto Lluch Lafuente (DTU, Denmark), and Andrea Vandin (IMT Lucca, Italy)

QUANTICOL meeting FMT – PisaMo – BicinCittà
ISTI–CNR, Pisa
20 January 2017
Outline

- Bike-sharing systems and bikes seen as product lines (configurable systems)
  - Multi-objective optimization with CLAFER

- Performance analysis of bike-sharing system (re)configurations and behavior
  - Mean field model checking with FLYFAST
  - Statistical model checking with QFLAN
Deciding a bike-sharing system (BSS) for a city poses many questions

- How many/what kind of bikes (features like light, basket, engine)?
- How many/what kind of stations (capacity), where to place them?
- How to avoid stations being completely full or empty for periods?
- Which BSS features (like maintenance, antitheft, smart services)?
- With or without dynamic redistribution of bikes? And what kind?
- Incentives (rewards) for users bringing bikes to less popular stations?
- Subscription costs and charging policy (like credit card or keycard)?

How to evaluate the numerous options, costs/benefits, improvements and changes in a systematic way? (i.e. performance, behavior)
Can we develop suitable decision support tools?
Product line (PL)

Product: a valid combination (configuration) of features

Product line or family: a set of valid feature combinations of a domain

Software Product Line Engineering (SPLE): Develop and maintain a (software) PL using a shared architecture or platform (commonalities) and mass customization (variabilities) to serve, e.g., different markets, thus allowing for (software) reuse

Aim: Maximize commonalities whilst minimizing cost of variations (i.e. of individual products)
Variability in terms of **features and constraints:**

- stakeholder visible pieces of functionality of a system . . .
- . . . which may be optional and/or may have alternatives
- only specific feature combinations lead to valid products

**Feature model:** compact representation of all valid products of a PL
Scalability is a major issue!

(slides by C. Kästner, Carnegie Mellon University, USA)
 atención feature model of a BSS

\[
\text{product: subset of features satisfying all variability constraints}
\]

\[
\text{cost}\left(\text{product}\right) = \sum\{ \text{cost}\left(\text{feature}\right) \mid \text{feature} \in \text{product} \}
\]
**ClaferMoo Visualizer:** compare different system configurations (product variants) w.r.t. various quality dimensions, select the most desirable variant, possibly by resolving trade-offs, and understand the impact of reconfigurations on a variant’s quality dimensions.

(minimizing cost whilst maximizing customer satisfaction and capacity)
Performance analysis BSS configuration

Assess performance of a BSS with mean field model checker **FLYFAST**
(developed by Latella & Massink together with Michele Loreti (UNIFI))

**Configuration 1** 330 stations: 100 of type **Periphery** with capacity 5
150 of type **Middle** with capacity 10
80 of type **Centre** with capacity 15

![Graph showing the average filling degree of docking stations over time for different configurations](image)
Performance analysis of reconfiguration

Configuration 2  390 stations: 200 of type Periphery with capacity 5  
150 of type Middle with capacity 10  
40 of type Centre with capacity 15

While P-type stations are in general less empty in configuration 2, 
filling degree of C-type stations fluctuates more in configuration 2.
What’s the probability that, within 30 mins, a peripheral station gets full, but then, with high probability, has a free slot within 6 mins, or doesn’t get full, but then, with low probability, gets full within 6 mins?
Additional feature constraints

C1 $\sum_{f \in P_F} \text{price}(f) \leq 600$: a bike may cost at most 600 €

C2 $\sum_{f \in P_F} \text{weight}(f) \leq 15$: a bike may weigh up to 15 kg

C3 $\sum_{f \in P_F} \text{load}(f) \leq 100\%$: a bike’s total computational load may not exceed 100%
Bike product line: behavioral constraints

Additional action constraints

C4 \( \text{do(sell)} \rightarrow \sum_{f \in \mathcal{P}_F} \text{price}(f) \geq 250 \)

C5 \( \text{do(irreparable)} \rightarrow \sum_{f \in \mathcal{P}_F} \text{price}(f) \leq 400 \)
Quantitative analysis Bike product line

Assess performance of Bike PL with statistical model checker QFLan
(developed by Vandin together with Ter Beek, Lluch Lafuente & Legay)

- $P_1$ Average price, weight or load of a bike
- $P_2$ For each feature, the probability to be installed
- $P_3$ The probability for a bike to be disposed of (irreparable)

When evaluated at a bike’s first deployment (for two versions of C1 + C2):

<table>
<thead>
<tr>
<th></th>
<th>Attributes ($P_1$)</th>
<th>Features ($P_2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>C2</td>
<td>price</td>
</tr>
<tr>
<td>600</td>
<td>15</td>
<td>391.91</td>
</tr>
<tr>
<td>800</td>
<td>20</td>
<td>509.83</td>
</tr>
</tbody>
</table>

- Probability of installing an engine ($g$) is very low, estimated at 0
  (i.e. with probability 0.9 it belongs to [0, 0.05], according to the confidence interval)

- This is likely because original C1 + C2 (first row) are too strict...
  (average price and weight is 391.91 € and 7.8 kg, but engine costs 300 € and weighs 10 kg)
For each feature, the probability to be installed over time:

\( P_2 \)

- Initial configuration \((P(y) = P(d) = 1)\)
- Pre-configuration (FACTORY)
- Customization (DEPOT)
Implementation as Eclipse-based tool

```
begin model BikesIDE3
  begin abstract features
    Bike
      Wheels Energy CompUnit Frame
    Tablet
      Energy Light Dynamo Engine Basket
    Engine
      MapApp NaviApp GuideApp Music
    GPS
    FirstStep
    end abstract features
  begin concrete features
    AllYear Summer Winter Light Dynamo Engine
    MapApp NaviApp GuideApp Music GPS Basket
    FirstStep
    Trashed
    end concrete features
  begin feature diagram
    Bike -> {Wheels, Light, Energy, Engine, CompUnit, Basket, Frame, FirstStep}
    Energy -> {Dynamo, Battery}
    CompUnit -> {Tablet, GPS}
    Frame -> {Diamond, StepThru}
    Tablet -> {MapApp, NaviApp, GuideApp, Music}
    end feature diagram
  begin feature predicates
    price = ([AllYear=100, Summer=70, Winter=80, Light=15, Dynamo=40, 
    Energy=0.3, Sumor=0.2, Winter=0.4, Light=0.1, Dynamo=0] 
    end feature predicates
  begin actions
    sell irreparable maintain book stop breakAction start assistance deploy 
    end actions
  begin constraints
```

MultiVoSTA client: at iteration 57 I have not reached all the required confidence intervals. I need the servers to perform a further batch of simulations (20).

MultiVoSTA client: still 2 queries to be evaluated

MultiVoSTA client: Overall number of performed simulation runs 1160

MultiVoSTA client: Batch 58 of simulations completed at 15:51:06.

MultiVoSTA client: analysis completed.

```
Publications acknowledging PisaMo/BicinCittà

[★: basis for this talk]


Publications which acknowledge PisaMo/BicinCittà
[*: basis for this talk]


- M.H. ter Beek, S. Gnesi, and F. Mazzanti, **Model Checking Value-Passing Modal Specifications.** In *Perspectives of System Informatics - Revised selected papers of the 9th International Ershov Informatics Conference (PSI’14).* LNCS 8974, Springer, 2015, 304–319.

QFLAN specification

\[ FR \doteq [ S \mid F ] \]

\[ S \doteq DS \ PS QS AS IS \]

\[ DS \doteq \ldots \ PS \doteq \ldots \ QS \doteq \ldots \ AS \doteq \ldots \ IS \doteq \ldots \]

\[ F \doteq (sell, 7).D \]

// Installing optional features:
+ (install(s), 6).F + (install(m), 10).F + (install(n), 6).F + (install(u), 3).F + (install(c), 20).F
+ (install(g), 4).F + (install(a), 5).F + (install(o), 10).F + (install(i), 10).F + (install(k), 8).F

// Replacing mandatory and exclusive features:
+ (replace(y, r), 5).F + (replace(y, w), 5).F + (replace(r, y), 10).F + (replace(r, w), 5).F
+ (replace(w, y), 10).F + (replace(w, r), 5).F + (replace(d, h), 3).F + (replace(h, d), 3).F

\[ D \doteq (deploy, 10).P \]

// Installing optional features:
+ \ldots same as F

// Uninstalling optional features:
+ \ldots same features and rates as installing, except for:
+ (uninstall(g), 1).D + (uninstall(a), 2).D + (uninstall(o), 3).D

// Replacing mandatory and exclusive features:
+ \ldots same as F, but Frame cannot be changed

// Replacing battery by dynamo in case no features requiring a Battery are installed:
+ (replace(a, o), 1).D

\[ P \doteq (book, 10).M + (maintain, 1).D \]

\[ M \doteq (stop, 5).H + (break, 1).B + (c, 20).M + (i, 20).M \]

\[ H \doteq (start, 5).M + (break, 1).B + (park, 5).P + (c, 20).H + (i, 10).H + (s, 10).H + (u, 10).H \]

\[ B \doteq (assistance, 10).D + (irreparable, 1).T \]

\[ T \doteq (install(trashed), 1).\emptyset \]
**FR** is composed of process $F$ and store $S$ of constraint sets:

**DS** Constraints from feature diagram (incl. cross-tree constraints)

$$DS \doteq y \otimes r \otimes w \ d \otimes h \ldots \ g \triangleright a \ n \triangleright m \ldots$$

**PS** Predicates for attributes of concrete features in feature diagram

$$PS \doteq \text{price}(y) = 100 \ \text{weight}(y) = 0.3 \ldots \ \text{price}(c) = 100 \ \text{load}(c) = 5 \ldots$$

**QS** Quantitative constraints

$$QS \doteq \text{price}(b) \leq 800 \ \text{weight}(b) \leq 20 \ \text{load}(b) \leq 100$$

**AS** Action constraints

$$AS \doteq \text{do}(\text{sell}) \rightarrow (\text{price}(b) \geq 250) \ldots \ \text{do}(c) \rightarrow \text{has}(c) \ldots$$

**IS** Initially installed features, i.e. **AllYear Wheels** + **Diamond Frame**

$$IS \doteq \text{has}(y) \ \text{has}(d)$$
A process in the specification

\[ FR \doteq [ S \mid F ] \]
\[ S \doteq DS \ PS \ QS \ AS \ IS \]
\[ DS \doteq \ldots \quad PS \doteq \ldots \quad QS \doteq \ldots \quad AS \doteq \ldots \quad IS \doteq \ldots \]
\[ F \doteq (\text{sell}, 7).D \quad \text{// Installing optional features:} \]
\[ + (\text{install}(s), 6).F + (\text{install}(m), 10).F + (\text{install}(n), 6).F + (\text{install}(u), 3).F + (\text{install}(c), 20).F + (\text{install}(g), 4).F + (\text{install}(a), 5).F + (\text{install}(o), 10).F + (\text{install}(i), 10).F + (\text{install}(k), 8).F \]
\[ \text{// Replacing mandatory and exclusive features:} \]
\[ + (\text{replace}(y, r), 5).F + (\text{replace}(y, w), 5).F + (\text{replace}(r, y), 10).F + (\text{replace}(r, w), 5).F + (\text{replace}(w, y), 10).F + (\text{replace}(w, r), 5).F + (\text{replace}(d, h), 3).F + (\text{replace}(h, d), 3).F \]

\textbf{F} implements \textit{FACTORY}’s behavior as a \textit{weighted} choice among:

1. With rate 7, the bike is sold and sent to depot. This action can only be executed if C4: \textit{do(sell)} \rightarrow (\textit{price(b)} \geq 250) is respected

2. Install optional features and iterate on \textit{F}. Such installations are only performed if \textit{DS} and \textit{QS} are preserved

3. Replace pre-installed mandatory exclusive features \textit{IS}, i.e. Frame or Wheels. Again, \textit{DS} and \textit{QS} must be preserved

\textit{QFLan}’s semantics forbids re-installing (installed) features