Title	Dynamic Adaptation of the Traffic Management System CarDemo
Authors(s)	Cordier, A.; Domingues, Rémi; Labaere, Anthony; Noel, Nicolas; Thiery, Adrien; Cerqueus, Thomas; Perry, Philip; Ventresque, Anthony; et al.
Publication date	2014-09-12
Conference details	2014 IEEE Eighth International Conference on Self-Adaptive and Self-Organizing Systems, SASO, London, UK, 8-12 September, 2014
Publisher	IEEE
Item record/more information	http://hdl.handle.net/10197/7211
Publisher's statement	© © 2014 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.
Publisher's version (DOI)	10.1109/SASO.2014.40

Dynamic Adaptation of the Traffic Management System CARDEMO

Arnaud Cordier*, Rémi Domingues*, Anthony Labaere*, Nicolas Noel*, Adrien Thiery*,
Thomas Cerqueus*, Siobhán Clarke‡, Pawel Idziak‡, Hui Song†, Philip Perry* and Anthony Ventresque*
* Lero@UCD, Performance Engineering Lab. School of Computer Science and Informatics
University College Dublin, Ireland. Email: {firstname.lastname}@ucd.ie
† SINTEF ICT, Norway, Email: hui.song@sintef.no
‡ Lero@TCD, Distributed Systems Group
Trinity College Dublin, Ireland. Email: {firstname.lastname}@tcd.ie

***Abstract*—This paper demonstrates how we applied a constraint-based dynamic adaptation approach on CARDEMO, a traffic management system. The approach allows domain experts to describe the adaptation goals as declarative constraints, and automatically plan the adaptation decisions to satisfy these constraints. We demonstrate how to utilise this approach to realise the dynamic switch of routing services of the traffic management system, according to the change of global system states and user requests.**

I. INTRODUCTION

CARDEMO is a Traffic Management System (TMS) developed to showcase research outputs produced within Lero[1]. The system offers all the basic functionalities of a TMS, e.g., viewing of a map, geolocation, routing services, localisation of Points Of Interests (POIs), viewing of traffic information. Most of these services use APIs and data provided by OpenStreetMap [1], [2]. Three types of users are defined: every-day user, public servant (e.g., a police officer) and TMS operator. As their expectations and needs are different, three distinct Graphical User Interfaces (GUIs) were developed. CARDEMO relies on the Entreprise Service Bus (ESB) software architecture model. In this type of architecture, functionalities are designed as services. Each service can be composed of other services, and services interact to form the whole system. Mule ESB [3] was chosen as the development framework for CARDEMO.

As the context of the application may change (e.g., a service may fail) and the requirements of the users may vary, a system needs to be able to adapt. In this paper we present an adaptation at runtime mechanism, and we show how it is integrated in a complex and real application. In this work, the adaptation mechanism is used to adapt a specific component of CARDEMO: the routing service.

II. ADAPTATION AT RUNTIME METHOD

A. General idea

The adaptation of routing methods of CARDEMO is implemented by the DYSARM adaptation tool [4]. The tool implements a constraint-based adaptation approach, as shown in Figure 1.

[1]The Irish Software Engineering Research Centre (www.lero.ie).

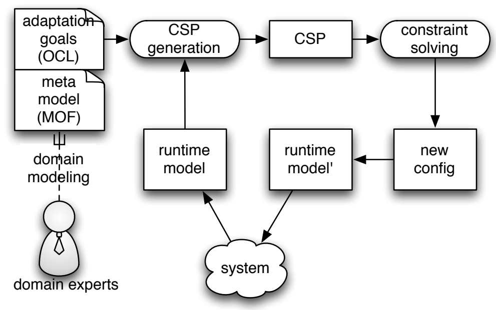

Fig. 1. Overview of DYSARM constraint-based adaptation.

The DYSARM adaptation tool requires domain experts to provide two parts of specification for the target domain. A *meta-model* defines what types of elements the system can be composed of in this domain, and the properties of each type. An *adaptation goal specification* describes the constraints of these types, or in other word, what is the expected system status in this domain. The two specifications are described in MOF (Meta Object Facility) and OCL (Object Constraint Language), respectively. At runtime, the tool uses the models@runtime technique to abstract the current system state into a runtime model conforming to the meta-model. From this model and the adaptation goals, it generates a constraint satisfaction problem, and uses constraint solving to find a new configuration of the system, which satisfies as many adaptation goals as possible. From the new configuration, the models@runtime engine calculates the difference and execute them back into the running system.

B. Integration in CARDEMO

Among all the features offered in the GUIs, this paper focuses on the routing. Table I lists the requirements (first 3 lines) and the features (last 4 lines) of six routing algorithms that are integrated in CARDEMO. The algorithms *osm2po-A** and *osm2po-Dijkstra* come from the osm2po API [5]. *OSRM* is provided through the OSRM API, while *Dynamic Routing* was developed to take into account the presence of events (e.g., car crash, protest) for the routing.

TABLE I
REQUIREMENTS AND FEATURES OF THE ROUTING ALGORITHMS.

	osm2po-A*	osm2po-Dijkstra	OSRM	Dynamic Routing	SUMO-Duarouter	SUMO-Dijkstra
Graph needed				✓		
SUMO needed					✓	✓
Only Dublin				✓	✓	✓
Considers events				✓		
Provides the shortest path	✓	✓				
Supports via points	✓	✓	✓			
Provides routing directions			✓			
Ping demand (access time)	3	2	1	5	4	3

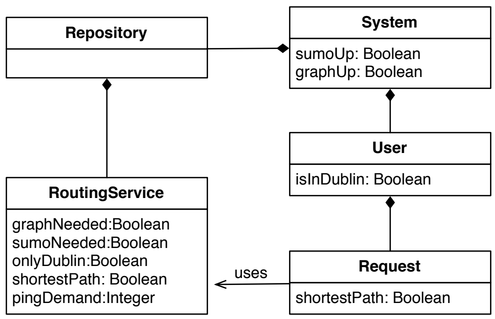

Fig. 2. Excerpt of the Meta-Model for CARDEMO.

Finally, *SUMO-Duarouter* and *SUMO-Dijkstra* come from the SUMO simulator [6].

Figure 2 shows an excerpt of the meta-model we defined for the TMS. The system maintains a repository of available routing services, and it carries global states, such as if the SUMO simulator is running. The system is monitored at regular intervals, allowing the capture of the global states changes at runtime. Similarly, different users, and even different requests from the same user, have different requirements. The adaptation problem is to compute and select the best candidate service to answer the requests according to the global state, and the requirement of the current request. The sample constraints listed below illustrate how to specify the adaptation goals. The first two constraints regulate that the current request needs a service that supports shortest paths or considers events; the selected service must provide one of these features. The third constraint utilises the global system state: the selected service can be *SUMO-Duarouter* or *SUMO-Dijkstra*, only if the current system has a SUMO simulator up and running. Finally, the last constraint expresses that the time required to access the service should not exceed the scope of pings allowed for the current user. For each constraint, we give a priority, and lower ones may not be satisfied in some circumstances. When the priority value is set to -1, it means that the constraint cannot be violated.

```
context Request inv: #priority = 3
   shortestPath implies uses.shortestPath
context Request inv: #priority = 5
   consideringEvents implies uses.consideringEvents
context Request inv: #priority = -1
   uses.sumoUp implies user.system.sumoUp
context Request inv: #priority = 2
   uses.pingDemand <= user.ping
```

The tool solves these constraints as follows. When the current state is `[system.sumoUp=false, user.ping=2, user.isInDublin=false]`, the solving result, and the adaptation decision, is to select *osm2po-Dijkstra*. For another example, if the request requires both shortest path and the consideration of events, than no candidate service can satisfy both, and the constraints are not satisfiable in this context. As the considering events has a higher priority, the result will be *Dynamic Routing*, ignoring the constraint related to shortest path.

III. DEMONSTRATION SCENARIOS

As it is not straightforward to simulate a realistic context of execution for the system, the changes of context are done artificially. Concretely, the GUI of CARDEMO includes a panel in which it is possible to disable a given service. For instance, a button allows to disable the SUMO service. In the demonstration we will show that the failure of the SUMO service, which represents a context change, triggers an adaptation of the system: the routing service is replaced. A video showing this specific scenario is accessible on the website of the CARDEMO project: `http://cardemo.ucd.ie/saso-demo.mov`.

ACKNOWLEDGMENTS

This work was supported, in part, by Science Foundation Ireland grant 10/CE/I1855 to Lero - the Irish Software Engineering Research Centre (www.lero.ie).

REFERENCES

[1] M. M. Haklay and P. Weber, "OpenStreetMap: User-Generated Street Maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, 2008.
[2] J. Bennett, *OpenStreetMap: Be your own Cartographer*. Packt Publishing, 2010.
[3] MuleSoft. (2014) Mule ESB: Most Popular Open Source ESB. [Online]. Available: http://www.mulesoft.com/platform/soa/mule-esb-open-source-esb
[4] H. Song, S. Barrett, A. Clarke, and S. Clarke, "Self-adaptation with end-user preferences: Using run-time models and constraint solving," in *ACM/IEEE 16th International Conference on Model-Driven Engineering Languages and Systems (MODELS)*. Springer, 2013, pp. 555–571.
[5] osm2po. (2014) osm2po: OpenStreetMap converter and routing engine for Java. [Online]. Available: http://osm2po.de
[6] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO - Simulation of Urban MObility: An Overview," in *3rd International Conference on Advances in System Simulation (SIMUL)*, 2011, pp. 63–68.