DeMOCAS: Domain Objects for Service-based Collective Adaptive Systems

Antonio Bucchiarone, Martina De Sanctis, Annapaola Marconi, and Alberto Martinelli

Fondazione Bruno Kessler, Via Sommarive, 18, Trento, Italy
{bucchiarone,msanctis,marconi,amartinelli}@fbk.eu

Abstract. DeMOCAS is a framework for the modeling and execution of service-based collective adaptive systems operating in dynamic environments. In this framework, we apply the Domain Object model and a Collective Adaptation algorithm to a case study from the mobility domain and we show its advantages in handling large-scale, decentralized and adaptive applications.

1 Introduction

Collective Adaptive Systems (CASs) are composed of distributed and heterogenous entities coming from both the real world and the back-end computer systems. Entities provide different system functionalities through their services. Services are defined as unique identifiable building blocks representing a concrete functionality in a larger, multi-service system. Implementing a functionality may involve interacting with other services through pre-defined protocols. At the same time, due to the high dynamism of the environment in which each service operates, the system must be able to re-configure its behavior at run-time in order to satisfy new users requirements and to fit new situations (i.e., unexpected and unlikely context changes). These systems with many services running in parallel, tend to display non-linear dynamics, which lead to decentralized adaptations that can break the consistency of the whole collaboration. Because of this, the adaptation must be itself collective, that is, multiple services must adapt simultaneously in a way that they properly address a critical runtime condition, while, at the same time, they preserve the collaboration and its benefits. In this paper we present **DeMOCAS** [1] a framework for the modeling and execution of service-based CASs. It includes mechanisms for services *specialization* and *adaptation* [3] and exploits the concept of *Domain Object* [2] [2] as a way to model customizable and adaptable services. DeMOCAS is build around three main aspects: (i) *dynamic settings*: each CAS is a collection of autonomous entities entering and exiting the system dynamically; (ii) *collaborative nature of systems*: entities can collaborate in groups (i.e., *ensembles*) for their mutual benefit; (iii) *collective adaptation*: multiple entities must adapt their behavior in concert to respond to critical runtime impediments. To demonstrate DeMOCAS in action, we use a real-world scenario from the urban mobility domain[3].

2 Application Domain and Scenario

The scenario refers to a *Multi-modal and Collaborative Urban Mobility System* (UMS), in a smart-city context. Its goal is that of synergistically exploiting the heterogeneous city services (e.g., transport, smart-card, online payment services), providing accurate, real-time, and customized mobility services, for the support of the whole travel duration. In the following, we present two reference storyboards from the scenario.

Storyboard 1. It is concerned with the normal execution of the UMS from the point of view of a user that needs to move around in the city. The system allows users to plan and execute a

[1] DeMOCAS is downloadable at the link: https://github.com/das-fbk/DeMOCAS

[2] The Domain Object model extended for the purpose of collective adaptation has been submitted to the main conference ICSOC2016, and it is under review.

[3] A video featuring the DeMOCAS solutions within the ALLOW Ensembles Project - http://www.allow-ensembles.eu/ can be viewed at the link: https://youtu.be/H0_LjptwZDg

journey, by providing a set of required information (e.g., departure and arrival time and places) and preferences (e.g., preferred transport means). The system arranges the set of suitable mobility alternatives, among which the user can choose the preferred one (e.g., the flexibus). At execution time, ensembles made by heterogeneous entities (e.g., the flexibus company, the route manager, the flexibus driver, the passengers) are dynamically organized. Although autonomous in their execution, they share common goals (e.g., being on time at the destination point).

Storyboard 2. It refers to how the system deals with context changes affecting its execution (i.e., unexpected changes in the environment, unpredictable behavior of the participants). Every participant of an ensemble can benefit from being part of it. The different entities can collaborate to reach the common good, in the cases in which adaptation needs arise (e.g., the flexibus route is blocked, a passenger is late to a pick-up point). If a collective adaptation is triggered, a decentralized and collective decision management strategy starts, where each entity does its best by offering its ability to handle specific problems. For instance, if the route is blocked, the driver can change the route path making a new plan, passengers can change their pick-up points to adapt to the new route, or, if this is not possible, the system can provide alternative solutions. Moreover, when an *intra-ensemble* adaptation can not be solved, *inter-ensembles* adaptation can be performed.

3 Collective Adaptation Approach

In this Section we present the approach for modeling and execution service-based CASs, as used in the implementation of DeMOCAS.

Application Model. DeMOCAS exploits a *design for adaptation* model, called Domain Object, defined with the purpose of modeling and executing service-based CASs. The model allows for the application of advanced techniques for dynamic adaptation (both *local* and *collective*), through specific constructs. Briefly, the system entities are modeled as DOs. Each DO implements its own behaviour (the *core process*), as well as the services (*fragments*) it provides. Both of them can be specified only partially, by defining *abstract activities* specified by their *goals*. At runtime, when the DO knows more about the context in which he is running, abstract activities are automatically refined with (one or a composition of) available fragments provided by other DOs. This also allows a DO to span its knowledge about the operational environment. Thus, the adaptive system results in a dynamic network of DOs. Moreover, over this dynamic network, *ensembles* are modeled as groups of autonomous DOs sharing common goals. For collective adaptation purposes, which are required to deal with adaptation needs spanning over the scope of a single DO, the model provides a set of *solvers* and *handlers*. While solvers model the ability of a DO to handle one or more issues, handlers are used to capture issues, during the nominal execution of a DO, and to trigger the appropriate solver.

Application Execution. As already introduced, DeMOCAS implements and shows the application of local and collective adaptations. The first one deals with the postponement of the concrete implementation of a service to the runtime phase (i.e., when the context is known). This kind of adaptation can be performed autonomously by a DO. The second, instead, is necessary to deal with unexpected changes in the environment and unpredictable behavior of the systems users and entities. For this kind of adaptation, a collaboration between DOs is required.

Local adaptation is performed through the refinement of abstract activities, by exploiting advanced techniques for dynamic and incremental service composition [1]. For instance, in Figure 1 we can see the process model of the `Urban Mobility System` DO, in which the abstract activity `UMS_CalculateTripAlternatives` is defined. The specification of this activity is left to the runtime, when the available multi-modal planners exposing services for journeys planner are known. In the process execution view, we can see that a fragment has been injected in place of the abstract activity, and it is under execution. Moreover, through the refinement process, ensembles are formed. In the mobility domain, for instance, an ensemble can be made by the driver of a bus, its specific route and the passengers.

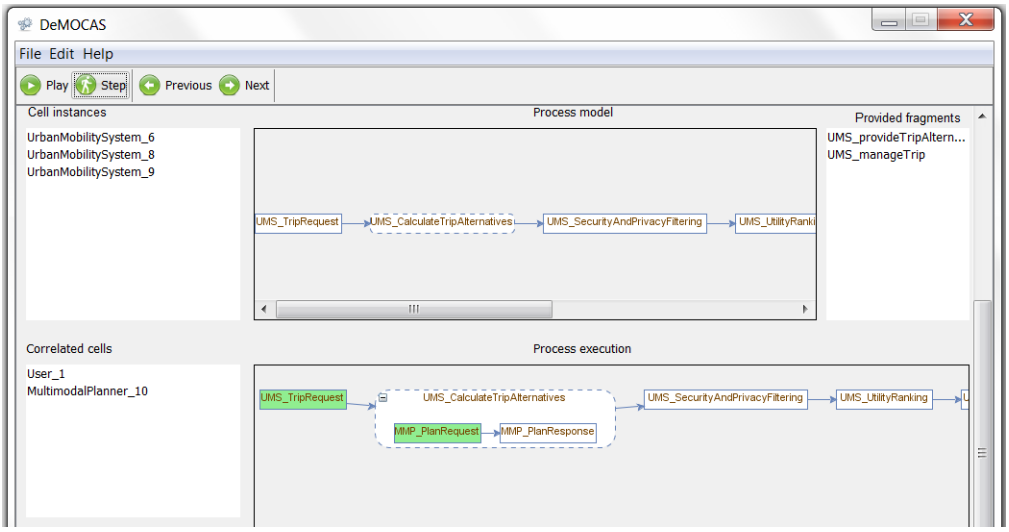

Fig. 1: Abstract Activities Refinement.

Collective Adaptation comes into play to handle unpredictable changes, which usually affect different running entities. As a consequence, adaptation must be collective, for making the system resilient and avoid a halt. The collective adaptation is performed by exploiting the handlers and solvers constructs, and by associating a MAPE (Monitor, Analyze, Plan, Execute) loop [4] to each DO. By monitoring the environment, handlers can both raise and–or catch issues. When an handler in a DO catches an issue, it calls the respective solver. A solver can solve the issue, or forward it outside the DO, if it is not able to provide a solution. In this case, a recursive procedure is triggered, leading to the construction of an issue resolution tree (as in Figure 2) in order to find a solution, if any. The chosen solution (the best one is selected in case of multiple solutions are available) will represent a path on the issue resolution tree. Moreover, it will be given by the union of the contributions coming for the solvers of the DOs involved in the resolution process. The best solution is eventually committed allowing the system to dynamically adapt and continue its execution. In Figure 2 we show the *Collective Adaptation Viewer* of DeMOCAS. The viewer reports the issue resolution result for the issue `Intense Traffic` triggered by a `Flexibus Driver`, during its route execution. In the left side, all the DOs involved in the issue resolution process are listed. The issue resolution tree of the `Route Manager` DO that owns the solver for the triggered issue, is shown in the right side.

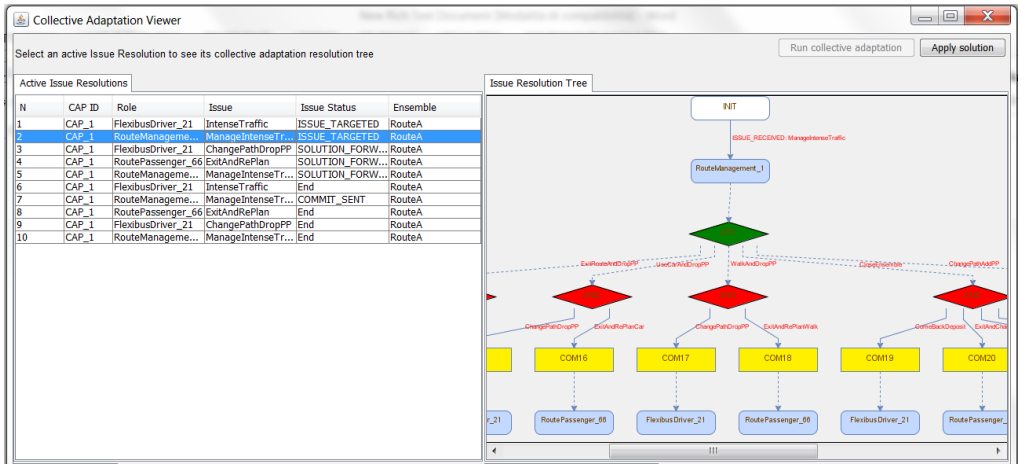

Fig. 2: Collective Adaptation Viewer of DeMOCAS.

4 Implementation

DeMOCAS has been implemented by using ***Java*** as programming language (to be executed, *Java 8* is required). Then, we used ***Eclipse*** as developing IDE and ***maven*** for the dependencies development and the project organization. From a design point of view, we report the DeMOCAS deployment diagram, in Figure 3. The demonstrator is composed by five modules, namely the ***Execution***, ***Adaptation***, ***AI Planning***, ***Presentation*** and ***Model*** modules.

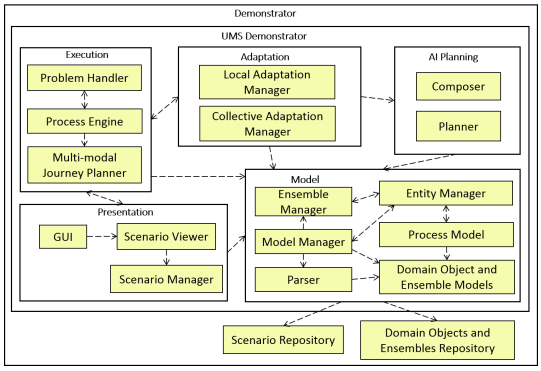

Fig. 3: DeMOCAS Deployment Diagram.

The ***Presentation*** module implements the DeMOCAS *GUI*, together with the functionalities allowing the scenario execution to be showed in the demonstrator interface. The graphical representation of the application domain is a map, on which the user can intuitively follow the evolution of the UMS scenario. The Presentation module is constantly synchronized with the Execution module, from which it obtains the details about the scenario execution.

In the ***Model*** module, the *Parser* is responsible for loading the scenario files (i.e., the Domain Objects and the Ensembles) from the file system and parse it. The Process Model defines the basic building blocks for a process definition (e.g., input, output and abstract activities), while the *Model*, *Entity* and *Ensemble Managers*, are essentially devoted to handle the models, entities and ensembles life-cycles.

The ***Execution*** module comes into play once the scenario, modeled in terms of DOs, has been loaded and displayed. The *Process Engine* is in charge of simulating the execution of the core processes of the running DOs instances, by performing all the corresponding activities. When runtime problems occur, the *Problem Handler* creates requests for the Adaptation module.

The ***Adaptation*** module is called by the Process Engine. When it must tackle the refinement of an abstract activity (*local* adaptation), as well as an issue resolution (*collective* adaptation), it uses the *Local Adaptation Manager* and the *Collective Adaptation Manager*, respectively. The Local Adaptation Manager is responsible for deriving the planning domain by driving the fragments selection and ranking (see [1]), according to the goal of the abstract activity and the specific execution context. The Collective Adaptation Manager runs the collective adaptation algorithm, for the issue that has been caught, by triggering the issue resolution procedure spanning different entities and ensembles.

The ***AI Planning*** module sustains the local adaptation of DOs, by supporting the refinement of abstract activities. It is in charge of managing the refinement procedure as an AI planning problem (*Planner*), providing the (composition of) fragments to be injected (*Composer*).

References

[1] A. Bucchiarone, A. Marconi, C. A. Mezzina, M. Pistore, and H. Raik. On-the-fly adaptation of dynamic service-based systems: Incrementality, reduction and reuse. In *Service-Oriented Computing - 11th International Conference, ICSOC 2013*, pages 146–161, 2013.

[2] A. Bucchiarone, M. De Sanctis, A. Marconi, M. Pistore, and P. Traverso. Design for adaptation of distributed service-based systems. In *Service-Oriented Computing - 13th International Conference, ICSOC 2015, Proceedings*, pages 383–393, 2015.

[3] A. Bucchiarone, M. De Sanctis, A. Marconi, M. Pistore, and P. Traverso. Incremental composition for adaptive by-design service based systems. In *IEEE 23rd International Conference on Web Services, San Francisco, USA, June 27 - July 2, 2016 (To Appear)*, 2016.

[4] IBM. An architectural blueprint for autonomic computing. Technical report, IBM, 2006.