Scalable Computing:
Practice and Experience

Scientific International Journal
for Parallel and Distributed Computing

ISSN: 1895-1767

Universitatea de Vest din Timişoara

Volume 15(4) December 2014

SUBSCRIPTION INFORMATION: please visit http://www.scpe.org

Scalable Computing: Practice and Experience

Volume 15, Number 4, December 2014

TABLE OF CONTENTS

Special issue on Enabling Technologies for Collaborations:

Introduction to the Special Issue **iii**

Engineering and Implementing Software Architectural Patterns Based on Feedback Loops **291**
Dhaminda B. Abeywickrama, Nicklas Hoch and Franco Zambonelli

Simulation Data Sharing to Foster Teamwork Collaboration **309**
Claudio Gargiulo, Delfina Malandrino, Donato Pirozzi and Vittorio Scarano

Investigation on the Optimal Properties of Semi Active Control Devices with Continuous Control for Equipment Isolation **331**
Michela Basili and Maurizio De Angelis

Regular Papers:

Optimizing Cloud Resources Allocation for an Internet of Things Architecture **345**
Bogdan Manaţe, Teodor-Florin Fortiş and Viorel Negru

DOI 10.12694/scpe.v15i4.1051
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 15, Number 4, pp. iii–iv. http://www.scpe.org

INTRODUCTION TO THE SPECIAL ISSUE ON ENABLING TECHNOLOGIES FOR COLLABORATIONS

Dear SCPE readers,

Collaboration is an important aspect in almost all fields of human life, and today the need for supporting collaboration is increased by the fact that we are always connected by means of different kinds of devices. In particular in the enterprise world, this need has emerged and satisfying it can lead to relevant benefits for companies. In the last years, enabling technologies have evolved to meet new and challenging requirements. The aim of this special issue is to provide a selection of the state of the art, emerging trends, new technologies and best practices in the field of technologies that enable collaboration. The idea was born at the 2014 IEEE WETICE Conference on Enabling Technology: Infrastructure for Collaborative Enterprises, but the call was open to any submission.

This special issue features three articles that concern technologies that enable collaboration.

The first one, "Engineering And Implementing Software Architectural Patterns Based On Feedback Loops" by Dhaminda B. Abeywickrama, Nicklas Hoch and Franco Zambonelli, focuses on collaboration in decentralized system of autonomous service components. The paper proposes an Eclipse plug-in called SimSOTA, which supports the design and the implementation of self-adaptive systems. Different phases of the development are covered in a model-driven fashion based on self-adaptive architectural patterns.

The second one, "Simulation Data Sharing to Foster Teamwork Collaboration" by Claudio Gargiulo, Delfina Malandrino, Donato Pirozzi and Vittorio Scarano, addresses the collaboration among engineers involved in Computational Fluid Dynamics (CFD) simulations. A Web-based system is presented, called Floasys, which was developed starting from the experience in a big automotive company.

The third paper, "Investigation On The Optimal Properties Of Semi Active Control Devices With Continuous Control For Equipment Isolation" by Michela Basili and Maurizio De Angelis, faces the collaboration among devices, composing a single equipment. A control algorithm, derived from the Lyapunov method and adapted to the addressed problem is presented, which is able to achieve the optimal isolation properties of semi active variable stiffness devices with continuous control across the whole frequency spectrum.

It is interesting to remark that all the accepted papers present a strong connection with real requirements, but meet them by solid models or theory. We also hope that these papers can solicit new research directions in the field.

We would like to thank the editorial board of SCPE for the chance of arranging this special issue, and all the reviewers for their hard work.

Giacomo Cabri, Federico Bergenti, Marco Aiello, Sumitra Reddy and Ramana Reddy

DOI 10.12694/scpe.v15i4.1052
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 15, Number 4, pp. 291–307. http://www.scpe.org

ENGINEERING AND IMPLEMENTING SOFTWARE ARCHITECTURAL PATTERNS BASED ON FEEDBACK LOOPS

DHAMINDA B. ABEYWICKRAMA*, NICKLAS HOCH †, AND FRANCO ZAMBONELLI‡

Abstract. A highly decentralized system of autonomous service components consists of multiple and interacting feedback loops which can be organized into a variety of architectural patterns. The highly complex nature of these loops make engineering and implementation of these patterns a very challenging task. In this paper, we present SimSOTA—an integrated Eclipse plug-in to architect, engineer and implement self-adaptive systems based on our feedback loop-based approach. SimSOTA adopts model-driven development to model and simulate complex self-adaptive architectural patterns, and to automate the generation of Java-based implementation code. The approach is validated using a case study in cooperative electric vehicles.

Key words: architectural patterns, autonomic systems, software engineering, self-adaptive systems, simulation, model-driven development, Eclipse plug-ins

AMS subject classifications. 68N99

1. Introduction.

Software systems are becoming increasingly complex and decentralized, and called to function in highly dynamic, open-ended environments. Thus, developing, deploying and managing these systems with the required level of reliability and availability have been very challenging. As a consequence, new software engineering methods and tools are required to make these systems *autonomic* [10], i.e. capable of automatically tuning their behaviour and structure in response to the dynamics of the operational environment in a *self-aware* and *self-adaptive* way.

In the context of the ASCENS project (Autonomic Service Component Ensemble, *www.ascens-ist.eu*), various models and mechanisms have been studied to integrate autonomic features in large-scale service systems, both at the level of service components and service ensembles. In particular, the SOTA ("State Of The Affairs") model has been defined [2] as a general goal-oriented framework for analysing the self-awareness and self-adaptation requirements of adaptive systems and for supporting the design of autonomic service ensembles.

To achieve such autonomic capability, *feedback loops* are required inside the system. The SOTA framework accordingly defines a catalogue of *self-adaptive patterns*, defining the many possible ways according to which feedback loops can be organized [6, 13]. However, a highly decentralized autonomic system may consist of a large number of service components, and multiple autonomic managers that close multiple and interacting feedback loops. Therefore, in addition to the catalogue, a framework such as SOTA should also provide solid engineering tools to actually support the process of designing and implementing autonomic systems that integrate such patterns. Several works like [8, 11, 12, 14, 15, 16] have addressed the need to make feedback loops explicit or first-class entities. However, little attention has been given to providing solid tool support for their engineering and implementation.

In this paper, we present SimSOTA—an integrated Eclipse plug-in we have developed to architect, engineer and implement self-adaptive systems based on our feedback loop-based approach. The SimSOTA integrated plug-in contains several plug-ins grouped together (i.e. a simulation plug-in and plug-ins for transformation). Our model-driven engineering approach integrates both decentralized and centralized feedback loop techniques in order to exploit their benefits. The SimSOTA plug-in facilitates (1) the modelling, simulating and validating of self-adaptive systems based on the SOTA feedback loop-based approach; and (2) the automatic generation of pattern implementation code in Java using transformations. We validate and assess our approach and plug-in using a case study in cooperative electric vehicles (e-mobility) [9].

In our previous work [4, 5], we presented early results of SimSOTA as a simulation plug-in for modelling and simulating patterns. However, the initial plug-in did not facilitate any application-independent instantiation of models or implementation of complex feedback loops, as supported by the current integrated Eclipse plug-in

*Fraunhofer FOKUS, Berlin, Germany (dhaminda.abeywickrama@gmail.com).
†Corporate Research Group, Volkswagen, Wolfsburg, Germany (nicklas.hoch@volkswagen.de).
‡Dipartimento di Scienze e Metodi dell'Ingegneria, Universit degli studi di Modena e Reggio Emilia, Italy (franco.zambonelli@unimore.it).

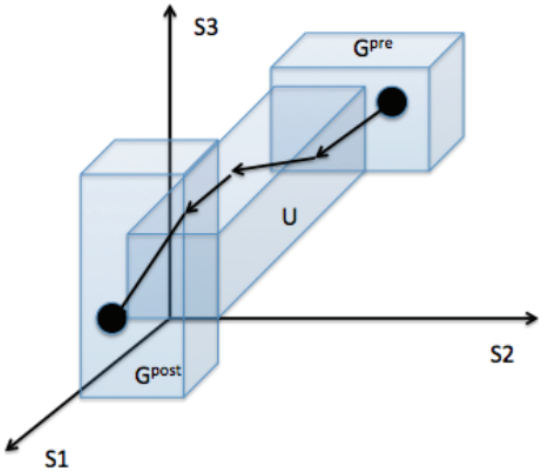

FIG. 2.1. *The trajectory of an entity in the SOTA space [2].*

of SimSOTA. The current paper extends [1] with more results of patterns engineered and implemented. This paper is extended with the help of a detailed case study in e-mobility showing the feasibility of the approach in complex scenarios. An additional pattern called the `Parallel AMs SC` pattern is introduced. The SOTA patterns profile created to represent our feedback loop-based approach is also introduced here. Another key addition is a detailed discussion of our approach against other key works.

The rest of the paper is organized as follows. In Section 2, we present the SOTA conceptual model and patterns catalogue, and the motivation to provide tool support. Section 3 describes the domain independent models created to facilitate the engineering process of the patterns. The case study used to derive platform-specific models is explained in Section 4. In Section 5, the domain-specific models created for the case study at both platform-independent and platform-specific levels are discussed. Section 6 provides a discussion of our approach and related work. Section 7 concludes this paper.

2. SOTA Patterns Catalogue and Tool Support. The SOTA model defined in the context of the ASCENS project considers that the actual execution of service components (SCs) and ensembles (SCEs) can be assimilated to that of a dynamical system: during its execution, it traces a trajectory in a virtual n-dimensional state space whose dimensions represent the relevant parameters of the execution, and where the goal G_{post} of system execution is reaching a given "state of the affair", i.e. a specific area in the state space, starting from a given initial area G_{pre} (the precondition for activating the goal), and possibly following a bounded trajectory U. Figure 2.1 represents these concepts graphically. Accordingly to such perspective, a system is considered self-aware if it can autonomously recognize its current position and direction of movement in the state space, and self-adaptation means that the system is able to dynamically direct its trajectory within U and towards G_{post} despite the fact that the dynamics of the operational environment tend to move it away from it. For a more formal description of these notions of SOTA refer to [2].

The SOTA modelling approach is key to understanding and modelling self-awareness and adaptation, and for checking the correctness of the specification [2, 3]. However, when a designer considers the actual architectural design of the system, it is important to identify which architectural schemes need to be chosen for the individual SCs and SCEs. To this end, To achieve such autonomic capability, *feedback loops* are required inside the system. That is, components and ensemble should somehow integrate control systems (in the form of so called "autonomic managers") to detect the current trajectory of the executing system, and when needed correct it so that specific regions of the space can be reached which correspond to specific application goals. The SOTA framework accordingly defines a catalogue of *self-adaptive patterns*, defining the many possible ways according to which feedback loops can be organized [6, 13], and supporting designers in their choices.

The SimSOTA tool completes the SOTA framework by supporting the actual process of designing and implementing such patterns, i.e. properly structured systems of SCs, SCEs, and their associated autonomic

managers. More in particular, SimSOTA (developed using IBM Rational Software Architect Simulation Toolkit 8.0) is an Eclipse plug-in aimed at providing an integrated environment for the architect to engineer and implement SOTA patterns in their respective domains. It contains several plug-ins grouped together, i.e. a simulation plug-in and plug-ins for transformation.

A key goal of SimSOTA simulation plug-in is to facilitate the engineering (modelling, simulating and validating) of complex self-adaptive systems based on feedback loops. It aims to provide a set of pattern templates (custom profile applied) for all the key SOTA patterns. This is to facilitate general-purpose and application-independent instantiation of models for complex systems based on feedback loops. The SimSOTA plug-in can also be used to simulate and animate the patterns to better understand their complex and dynamic model behaviour. Also, the feedback loop models and their interplay can be validated to detect errors early and to check whether the specified behaviour works as intended.

On the other hand, the transformation plug-ins of SimSOTA are aimed at facilitating the automatic generation of implementation code in Java for the patterns. SimSOTA applies model transformations to automate the application of UML-based architectural design patterns and generate infrastructure code for the patterns using Java semantics. Here, model transformation techniques are applied as a bridge to enforce correct separation of concerns between two design abstractions, i.e. UML-based patterns and their Java implementations. Two main benefits of applying transformations here are improving the quality and productivity of patterns development, and easing system maintenance and evolution for the patterns engineer.

3. Domain-Independent Pattern Models. The models and code developed using the SimSOTA tool can be at the domain-independent or domain-specific levels. This section discusses the domain-independent pattern template models created to facilitate the model-driven engineering process of the patterns at the platform-independent UML and platform-specific Java levels. This is discussed using three key SOTA patterns of the catalogue, i.e. `Autonomic SC pattern`, `Parallel AMs SC pattern` and `Centralized SCE pattern`.

3.1. Notion of Feedback Loop used in SOTA. The notion of feedback loops explored in our patterns extends the well-established IBM's MAPE-K adaptation model [10] with multiple, interacting loops (see [4] for more details). MAPE-K (i.e. monitor, analyse, plan, execute over a knowledge base) is a reference model introduced by IBM for autonomic control loops [10]. Advanced software systems that are often highly decentralized require the modelling of multiple interacting control loops. Multiple feedback loops can coordinate and support adaptation using two basic mechanisms: *inter-loop* and *intra-loop* coordination [15]. Intra-loop coordination is provided by multiple sub-loops within a single feedback loop, which allows the various phases of MAPE-K in a loop to coordinate with one another. In contrast, inter-loop coordination supports the coordination of adaptation across multiple control loops. These inter-loops can interact using three basic mechanisms: *hierarchy*, *stigmergy* and *direct interaction* [4, 14].

Both decentralized and centralized feedback loop approaches have been suggested to facilitate autonomic behaviour in adaptive systems [4, 19]. We integrate these approaches in order to exploit the benefits of both. Although centralized approaches allow global behaviour control, they contain a single point of failure and suffer from scalability issues. Conversely, decentralized approaches do not require any a priori knowledge, nor do they contain a single point of failure.

We have developed a UML activity-based custom profile (`SOTA patterns profile`) to model the different elements of the SOTA feedback loop notion applied in the patterns. A custom UML profile introduces a set of stereotypes that extends the existing meta-model of UML to a specific area of application. The `SOTA patterns profile` extends and customizes the UML meta-model for activity diagrams described in the UML 2.4.1 infrastructure and superstructure specifications. It contains several stereotypes which are used by the transformation plug-ins when generating implementation code for the patterns.

SOTA identifies and classifies patterns based on the above described dimensions, and defines a taxonomy that is helpful for the engineering of multiple and possibly interacting feedback loops, in particular for choosing among a variety of feedback loop compositions.

3.2. UML Template Models for the Patterns. We provide a set of UML pattern templates for all the key SOTA patterns (e.g. `Primitive SC`, `Proactive SC`, `Autonomic SC`, `Parallel AMs SC`, `Multilevel AMs SC`, `Centralized SCE`, `P2P AMs SCE`, `Cognitive Stigmergy SCE`, `Hierarchy of AMs SCE patterns`). The

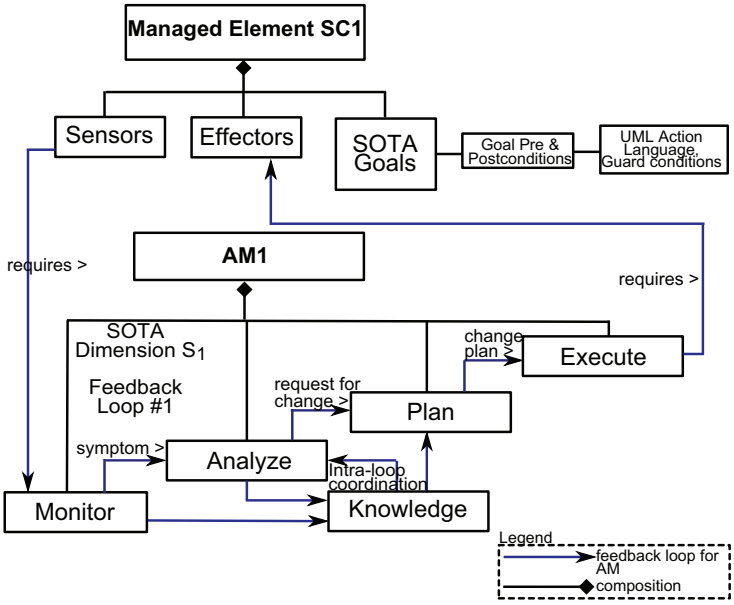

FIG. 3.1. *Autonomic SC pattern: Conceptual model.*

main goal behind this is to facilitate general-purpose and application-independent instantiation of models for complex systems based on feedback loops. More specifically, it [1]:

(i) provides the engineer a starting structure for pattern modelling activities in support of capturing details related to patterns modelling. The templates are used to specify any architecturally significant structures that need to be included in the activity-based pattern models created using the templates.

(ii) presents useful guidance and textual advice to the engineer on applying the profile and deriving platform-specific models in a consistent manner. It can provide instructions to the engineer on how to fill and complete the model using elements within the template, and using features of the tool environment.

For a high-level and conceptual description of the key SOTA patterns of the catalogue refer to [13]. At the SC level, these patterns use a decentralized feedback loop approach while at the SCE level, they primarily use a centralized feedback loop approach. Thus, our work applies both decentralized and centralized feedback loop techniques in order to exploit their benefits. Although these template models are not Eclipse plug-ins, they are distributed in *plug-ins*. Here, we present UML template models created for three key SOTA architectural patterns at the SC (`Autonomic SC pattern`, `Parallel AMs SC pattern`) and SCE (`Centralized SCE pattern`) levels. The pattern templates have been modelled using UML 2.2 activity models (cf. Figs. 3.2, 3.4 and 3.6). The corresponding conceptual models of these patterns are provided in cf. Figs. 3.1, 3.3 and 3.5.

3.2.1. Autonomic SC Pattern. The `Autonomic SC pattern` is characterized by the presence of an explicit, external feedback loop to direct the behaviour of the managed element (cf. Figs. 3.1 and 3.2). The managed element has sensors, effectors and a representation of SOTA goals. The SOTA utilities are enforced in the managers. An *autonomic manager (AM)* handles the adaptation of the managed element.

In general, an SC and its manager model the following feedback loop behaviour [1]. The sensors in the SC capture event sensor data. This is then collected by the monitor phase of the manager, which filters and accumulates the event data. The event data is stored in the knowledge base component of the loop, which also stores predicate rules for the SOTA utilities. Then, the analyse phase of the loop gathers the event signals and utilities from the knowledge base component of the loop and interprets against the patterns. The result of this interpretation (event symptoms) is stored in the knowledge base component of the loop. The plan phase of the loop obtains event symptoms and interprets them. If the awareness level is not satisfied then it triggers and devises a plan to execute awareness change. To achieve this, the execute phase of the loop notifies the SC effector which adapts the required awareness level inside the SC accordingly.

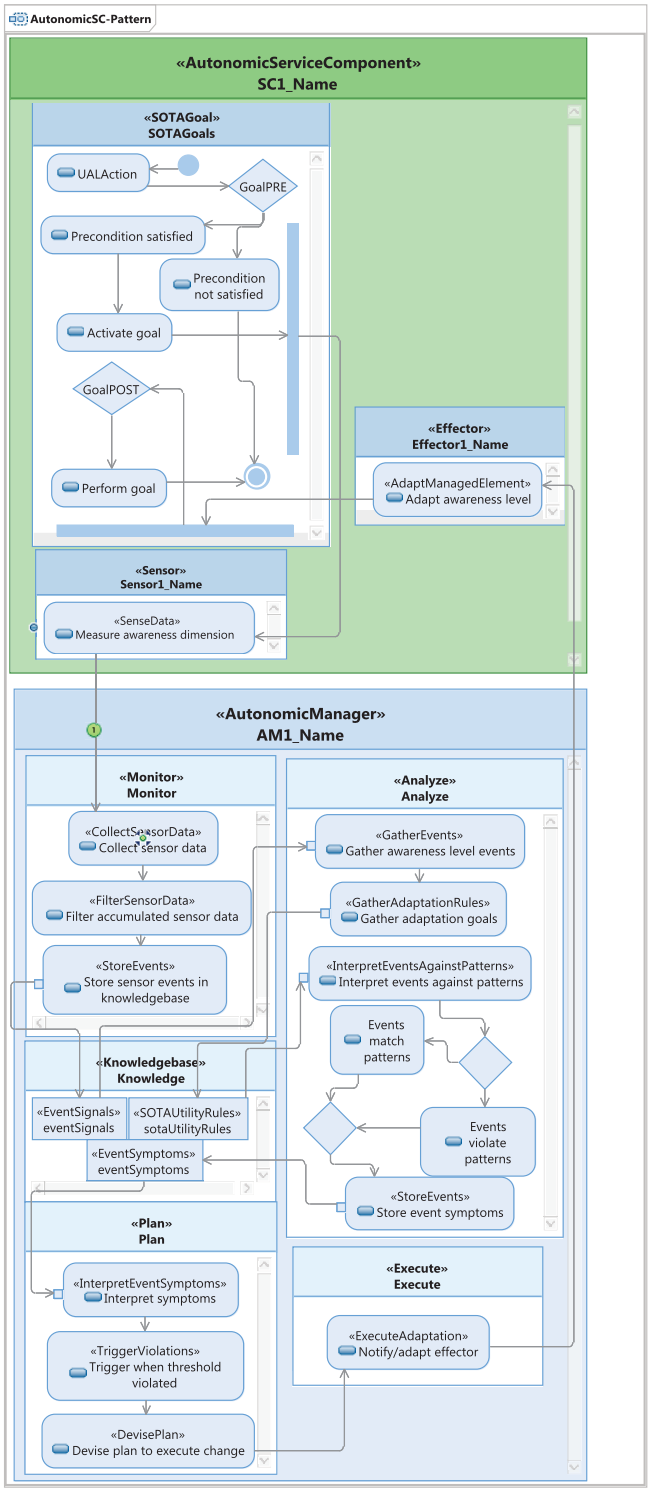

FIG. 3.2. *Autonomic SC pattern (continued): UML pattern template model.*

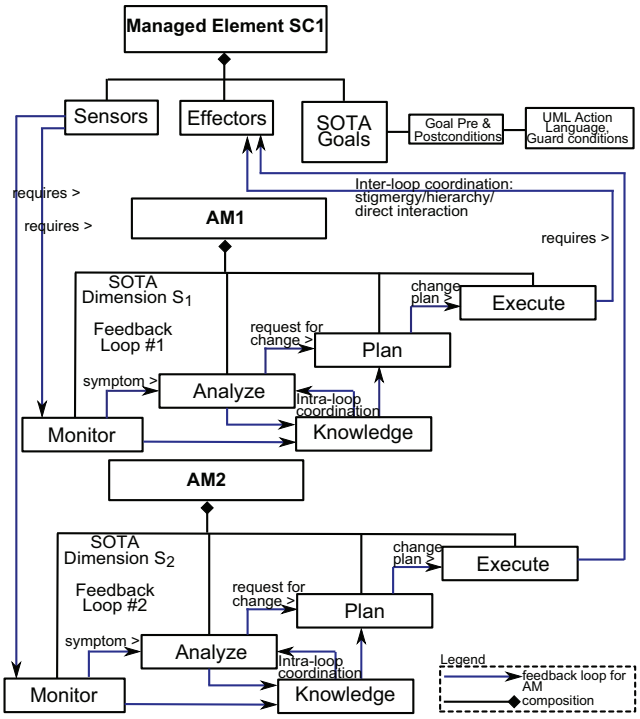

FIG. 3.3. *Parallel AMs pattern: Conceptual model.*

3.2.2. Parallel AMs SC Pattern. The `Parallel AMs SC pattern` is an extension of the `Autonomic SC pattern` [13]. Here, several AMs can be associated with the managed element, each closing a feedback loop devoted to controlling a specific adaptation aspect of the system (cf. Figs. 3.3 and 3.4). Adding different levels of AMs increases the autonomicity, and these extra AMs work in parallel to manage the adaptation of the managed element. For instance, let us consider that we have two feedback loops with an AM in each loop to handle adaptation in two SOTA dimensions. These loops can interact with each other using *hierarchy*, *stigmergy* or *direct interaction*, and here we can identify an *inter-loop* coordination where MAPE-K computations of the loops coordinate with each other. Also, an *intra-loop* can be identified between the Analyse and Knowledge components of an AM to allow the coordination of adaptation between these two phases.

3.2.3. Centralized SCE Pattern. The `Centralized SCE pattern` is characterized by a global feedback loop, which manages a higher-level adaptation of behaviour of multiple autonomic components (cf. Figs. 3.5 and 3.6) [1]. The adaptation in the `Centralized SCE pattern` is handled by a *super AM* which is a high-level AM. Like an AM, a super AM has the MAPE-K adaptation model. This is while the single SCs are able to self-adapt their individual behaviour using their own external feedback loops. The feedback loops in this pattern can interact using *hierarchy*, *stigmergy* or *direct interaction*.

3.3. Java Templates for the Patterns. We provide a set of domain-independent templates in Java for the key SOTA patterns [1]. These domain-independent templates can be used as examples by the developer when implementing the patterns in their domains. An alternative solution can be to provide a reusable library for the key SOTA patterns. However, it is impractical to generalize programming patterns such that they could be reused across different complex systems based on feedback loops.

We have used the *Fork/Join framework* of Java SE 7 to implement the SOTA patterns that have `autonomic SCs` as the constituent SCs. The implementation of SOTA patterns that have other types of SCs as constituent SCs (e.g. `primitive SCs` and `proactive SCs`) is currently work in progress. In SOTA, *goals* represent the eventual state of the affairs that a system or component has to achieve [2]. On the other hand, *utilities* are constraints on the trajectory or execution path that a system should try to respect while achieving the goals.

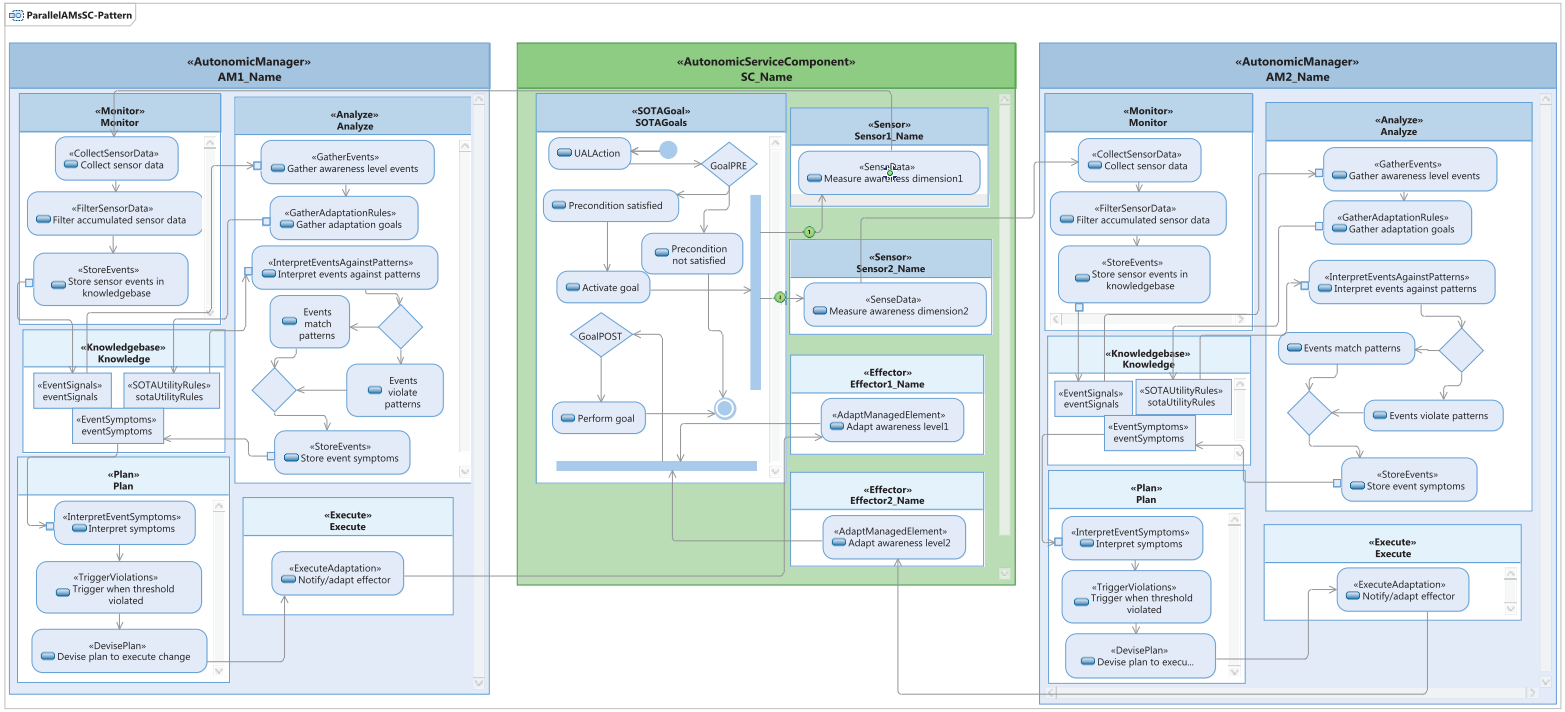

FIG. 3.4. *Parallel AMs pattern (continued): UML pattern template model.*

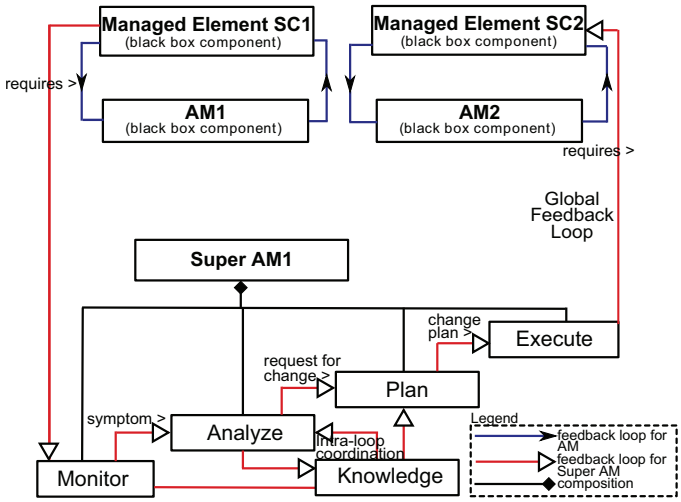

FIG. 3.5. *Centralized SCE pattern: Conceptual model.*

In the SOTA patterns that have `autonomic SCs` as the constituent SCs, the goals are performed by the SCs while utilities are handled by their respective autonomic managers. This design principle maps and corresponds well to the *Fork/Join Framework* of the Java SE 7, which has introduced the notion of *parallelism*. Using the Fork/Join framework, an SC delegates its awareness and adaptation handling activities (utilities) to its AMs. Utilities are enforced parallel to the execution of an entity in the SOTA space.

Each Java template (e.g. cf. Figs. 3.7 and 3.8) maps well to their corresponding UML design templates discussed previously. In all the templates, in general, an AM or a super AM has methods to handle the monitor, analyse, plan and execute phases of the feedback loops (e.g., `monitor()`, `analyse()`, `plan()`, `execute()` methods in cf. Fig. 3.7). HashMaps have been created to deal with the knowledge base elements of the loops (e.g. a HashMap called `am1EventSignals` to store event signals in the Knowledge base). On the other hand, in the SCs, the preconditions and the postconditions of the SOTA goals have been implemented as predicate rules. Also, an SC has methods to handle the behaviour of its sensors and effectors. As for the interactions, the AMs can implement behaviour for *intra-loops*, for example, the analyse method handling the analyse phase of the loop queries and stores data in the HashMaps, simulating an intra-loop behaviour. Meanwhile, the behaviour for *inter-loops* (i.e. *stigmergy*, *hierarchy* and *direct interaction*) have been implemented as method calls. Stigmergy has been implemented implicitly where two AMs communicate with each other through the SC they are connected with (e.g. `Parallel AMs SC pattern`).

For example, the Java template for the `Parallel AMs SC pattern` (cf. Fig. 3.7) has two AM classes associated with a single SC class, each closing a feedback loop controlling a specific adaptation aspect of the system. On the other hand, the template provided for the `Centralized SCE pattern` (cf. Fig. 3.8) has two SC classes and a super AM class. Out of these patterns, the `Autonomic SC` and `Parallel AMs SC` patterns exhibit a decentralized feedback loop approach, while the `Centralized SCE` pattern demonstrates a centralized feedback loop approach.

4. Case Study Scenario. This section describes the e-mobility case study problem used to derive platform-specific models in the pattern engineering and implementation process. The case study problem addresses the SOTA model's self-awareness and self-adaptation mechanisms.

The case study scenario concerns individual planning and mobility for a single user and a privately owned vehicle. Let us consider a situation where a user intends to travel to an appointment at a particular destination (cf. Fig. 4.1, top) [9]. First, the user drives the electric vehicle (e-vehicle) to the car park, then parks the car and walks to the meeting location. During the walking and meeting times, the e-vehicle can be recharged. Here, the main SCs are the user or driver, the e-vehicle, and the parking lots and charging stations (infrastructure). A main SCE in this scenarios is the temporal orchestration of the user, the e-vehicle, a parking lot and a charging

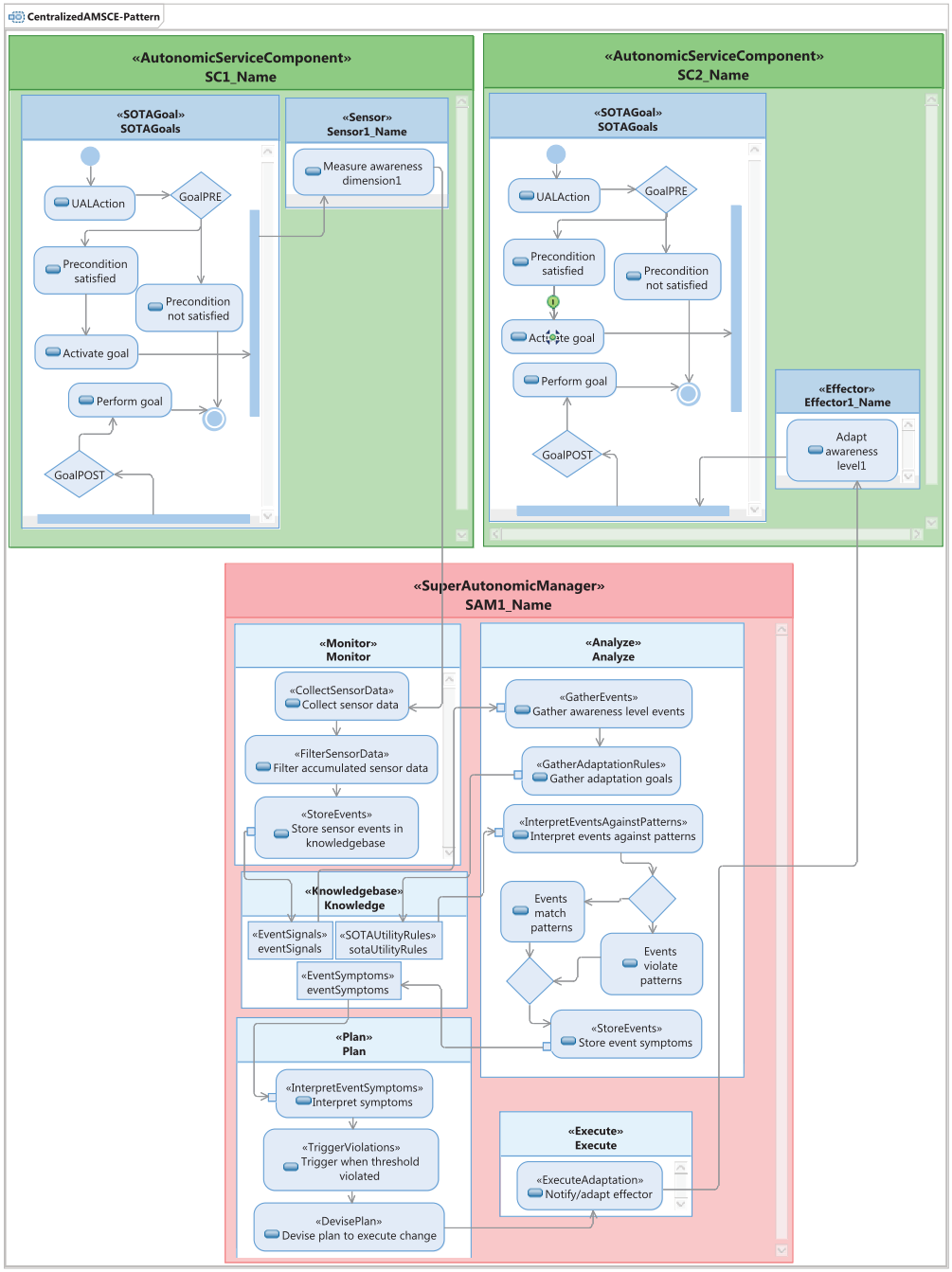

FIG. 3.6. *Centralized SCE pattern (continued): UML pattern template model.*

station assigned at trip start (*User-E-Vehicle-Assigned-Infrastructure SCE*, cf. Fig. 4.1).

The e-mobility case study problem is well suited to be solved using appropriate self-awareness and self-adaptation mechanisms (e.g. the SOTA model) for the following reasons [4]:

(i) The case study problem involves a significant number of SCs. Therefore, relying only on centralized solutions becomes non-feasible and providing for decentralized solutions to handle localized decision making is

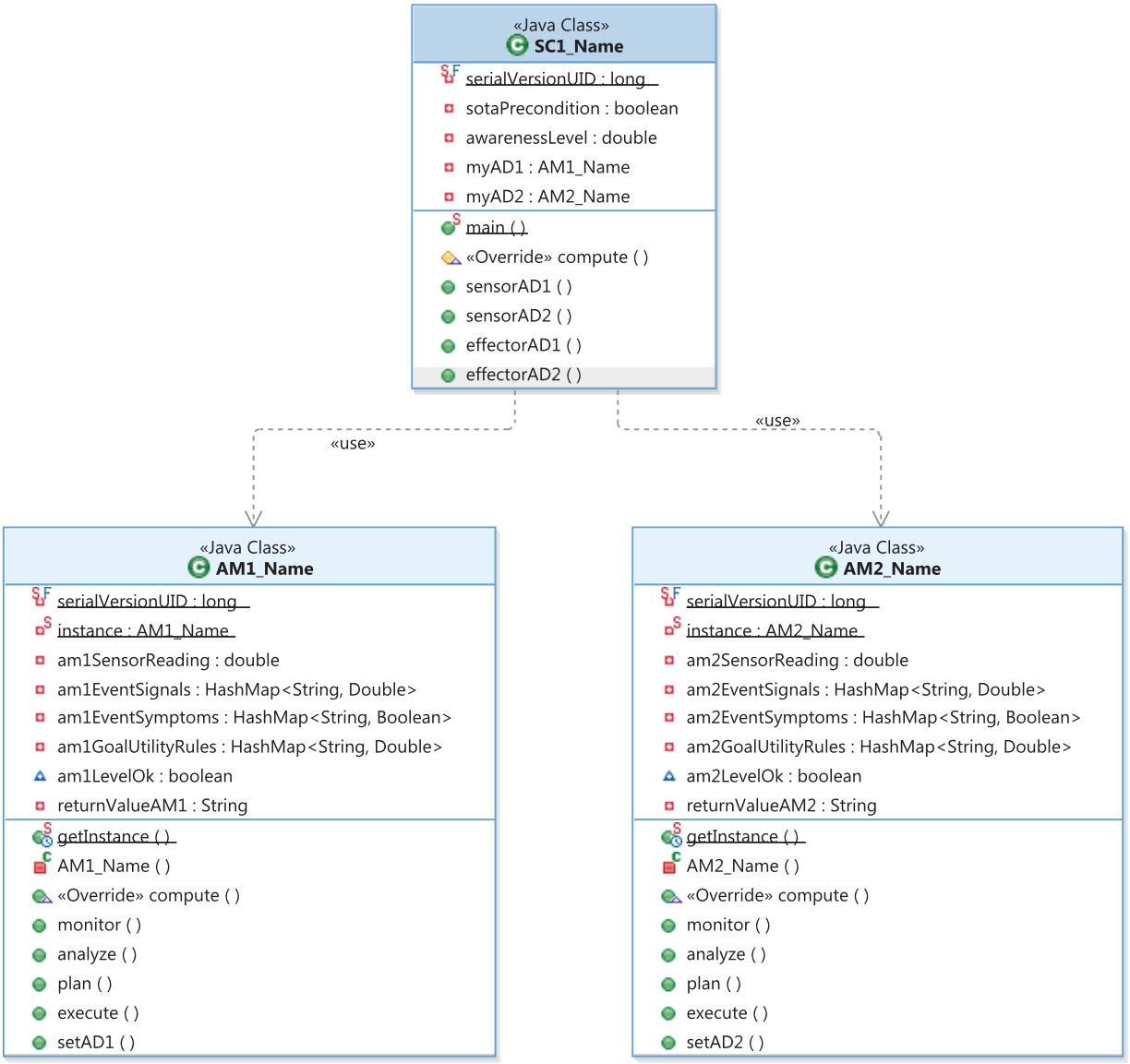

FIG. 3.7. *Java pattern template models: Parallel AMs SC pattern.*

important.

(ii) This problem deals with several awareness dimensions that need to be handled by the complex SCs, e.g. e-vehicle SC: time, energy, location; user SC: user preferences (climate comfort, driving style), time, cost; parking lots and charging stations SCs: availability.

(iii) Each SC has access only to partial information or partial view of the environment (e.g. during the trip, the e-vehicle is not aware of the availability of its assigned parking lot). Therefore, the individual SCs need strategies to adapt at run-time as more information becomes available to them.

(iv) Each SC and SCE is involved in significant levels of uncertainty or contingency situations (system or environmental changes) requiring self-adaptive actions. These can be (1) e-vehicle SC: the unavailability of a parking lot; the planned event deadline is missed (the e-vehicle could not reach the destination at the time required or with the energy planned); the user overrides the plan; (2) user SC: the shifting of an appointment;

FIG. 3.8. *Java pattern template models (continued): Centralized SCE pattern.*

and (3) parking lot and charging station SCs: the e-vehicle does not arrive at the booked time or it leaves earlier or later; charging is not initiated at the foreseen time and draws unforeseen amounts of power.

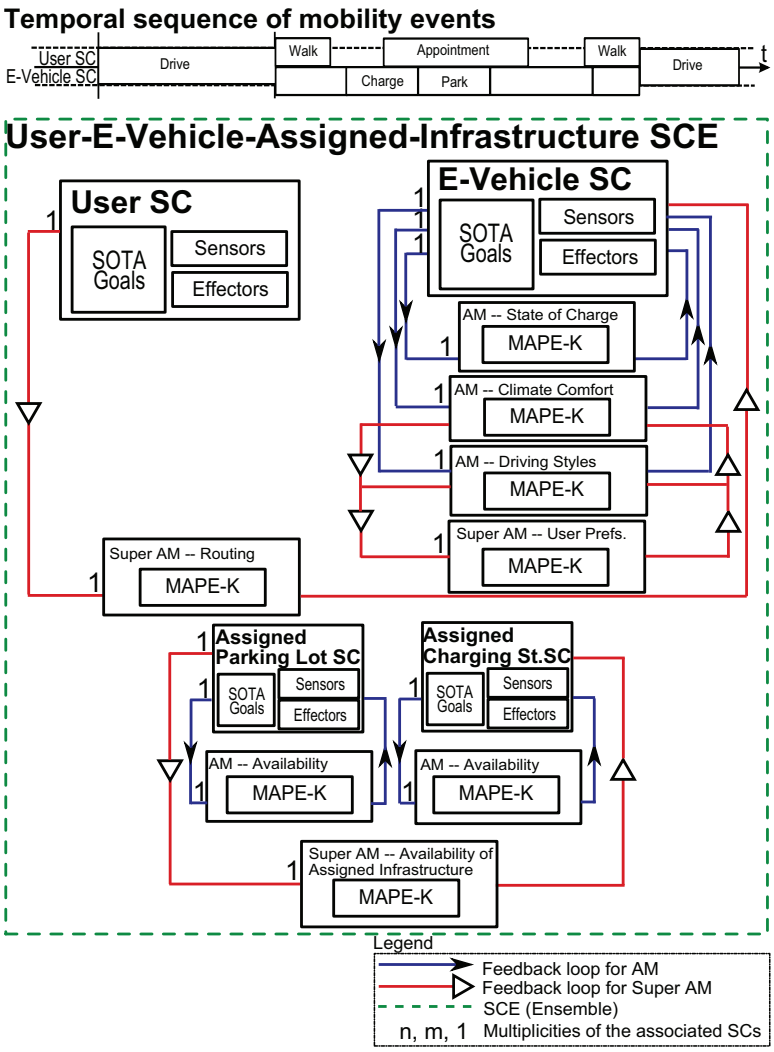

FIG. 4.1. *SOTA patterns applied to the e-mobility scenario.*

4.1. Case Study Scenario and Automation with SOTA. In the SOTA model, the SCs are conceptually modelled as entities moving and executing in the space, and these entities can have goals and utilities that describe how such goals can be achieved at an individual or a global level. Each SC and SCE of the mobility scenario can be described using the SOTA goals and utilities, the awareness being monitored for each managed element, and the self-adaptive behaviour using SOTA feedback loops as a response to any contingencies that may occur.

For example, let us consider the e-vehicle SC, which is the central SC within the mobility scenario. It interfaces with both the user and the infrastructure SCs during driving, and with the infrastructure SCs only during parking or walking. The goal of the e-vehicle is to reach the destination with the planned energy and at the planned time. A utility can be the constraint that the battery charge should not reach *low* until the e-vehicle reaches its destination. The monitored awareness dimensions are, among others, the state of the battery charge, temperature (for climate comfort requirements of the user), acceleration and velocity (for driving style requirements), and current location of the vehicle.

As shown in cf. Fig. 4.1, in order to handle the adaptation of a managed element, we can provide separate AMs (`Autonomic SC pattern` - Sect. 3.2.1) for each SOTA awareness dimension. The e-vehicle SC has three

AMs defined to handle the adaptation of the battery state of charge, climate comfort and driving style requirements of the user. Such separate AMs here provides an instance of the `Parallel AMs SC pattern` described in Sect. 3.2.2. Also, any self-adaptive behaviour on routing needs to be handled at the SCE level, as these actions are applicable to both the user and the e-vehicle SCs. Thus, a super AM has been defined to handle the adaptation of routing (`Centralized SCE pattern`). The parking lot SCs and charging station SCs need to be aware of their availability. To handle possible contingency situations, two separate AMs (`Autonomic SC pattern`) can be defined to manage the availability of the two assigned infrastructure SCs, and a super AM can be provided to handle the adaptation of both SCs (`Centralized SCE pattern`).

As seen here, it is clear that there are a number of SCs, SCEs, AMs and super AMs closing multiple, interacting feedback loops. These feedback loops, which SOTA uses as mechanisms to express self-awareness and self-adaptation, can be organized using several architectural patterns. Therefore, a key goal of our work is to provide engineering and implementation support to the software engineer in order to easily grasp this complex setup.

5. Domain-Specific Pattern Models. The domain-specific pattern models created at the platform-independent UML and platform-specific Java levels for the case study scenario are discussed in this section. Model transformations have been employed as a bridge to enforce correct separation of concerns between these two design abstraction levels.

5.1. Platform-Independent Pattern Models. We have used the simulation plug-in component of the integrated SimSOTA tool to instantiate the UML pattern template models for the e-mobility case study. UML 2.2 activity models have been used as the primary notation to model the behaviour of feedback loops. In a feedback loop, the actions are not necessarily performed sequentially. An iterative process allows the revision of certain decisions if required, and therefore activity diagrams are effective to design the feedback loops. The plug-in also facilitates the simulation of feedback loops in other UML 2.2 diagrams, such as composite structure and sequence diagram models [4].

We now briefly describe the domain specific, platform-independent models realized using SimSOTA to simulate a number of feedback loop structures in the *User-E-Vehicle-Assigned-Infrastructure SCE* of the e-mobility scenario (cf. Fig. 4.1). Note that due to space considerations, the activity model provided in cf. Fig. 5.1 is a subset of the *User-E-Vehicle-Assigned-Infrastructure SCE*. Here, we particularly describe the instantiation of three key SOTA patterns - `Autonomic SC pattern`, `Parallel AMs SC pattern`, and `Centralized SCE pattern`.

There are several managed elements or SCs: the e-vehicle (`SC_EVehicle`; cf. Fig. 5.1, top-center), the user (`SC_User`) and the assigned parking lot (`SC_ParkingLotAssigned`). Each managed element (e.g. e-vehicle SC) has an activity-based UML model to represent SOTA goals (e.g. reach destination) which can be characterized in terms of a precondition (e.g. whether the assigned parking lot is available) and a postcondition (e.g. actual reaching of the destination within the state of battery charge and time). The preconditions and postconditions of the SOTA goals are modelled using UML Action Language and guard conditions. The utilities for the e-vehicle SC are constraints on the state of the battery charge, climate comfort, driving style (acceleration and velocity), and routing requirements until the e-vehicle reaches its destination. The utilities are modelled in the managers.

5.1.1. Decentralized and Centralized Feedback Loops and Interactions. The SimSOTA simulation plug-in integrates both decentralized and centralized feedback control loop techniques to handle the adaptation of the managed elements. In this example, the decentralized feedback loop behaviour is provided by the `Autonomic SC pattern` and `Parallel AMs SC pattern`. Here, the `SC_EVehicle` and the AM on battery charge (`AM_EV_SoC`) form an instance of the `Autonomic SC pattern`. The `Sensor` components in `SC_EVehicle` form concurrent activities for measuring the battery charge, temperature (for climate comfort), location (for routing), acceleration and velocity (for driving style) inside the e-vehicle, respectively (cf. Fig. 5.1, top-center). These concurrent activities close several decentralized feedback loops in the AMs which interact using stigmergy and act on its shared subsystem. On the other hand, the `SC_EVehicle` and the two AMs on battery charge (`AM_EV_SoC`) and climate comfort (`AM_EV_ClimateComfort`) form an instance of the `Parallel AMs SC pattern` (cf. Fig. 5.1).

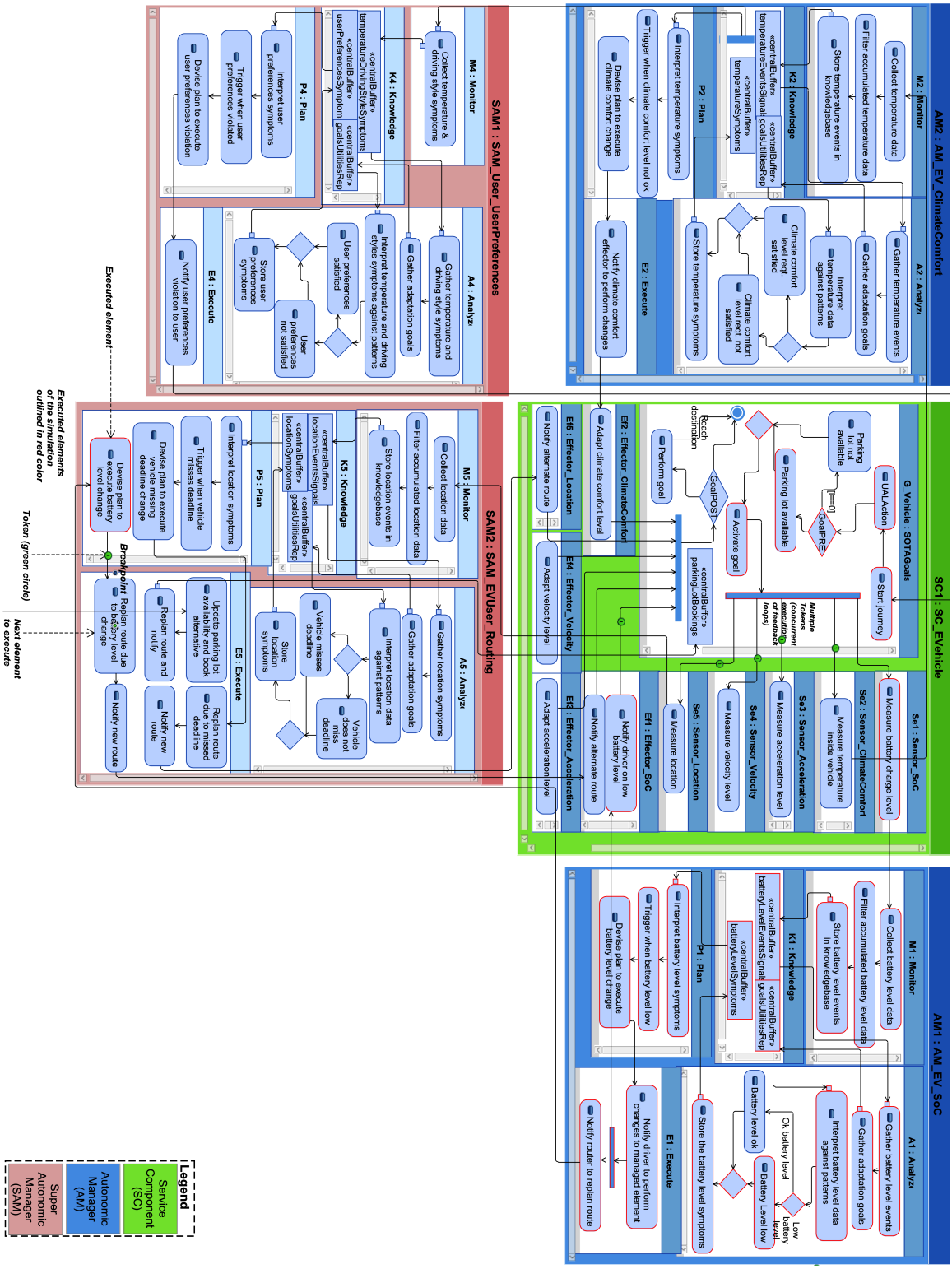

FIG. 5.1. *Patterns simulated as an activity model for an e-mobility scenario.*

The super AM defined for managing the adaptation of routing (`SAM_EVUser_Routing`; cf. Fig. 5.1, bottom-center) closes a separate, centralized (global) feedback loop. The super AM deals with both the user and the e-vehicle SCs. This instantiates the `Centralized SCE pattern`. The `SAM_EVUser_Routing` deals with contingency situations if the e-vehicle misses its deadline. Refer to [4] for more details on the domain-specific, platform-independent models (and interactions) derived in the example.

5.2. Transformations. Model transformations have been used in the current study to automate the generation of Java-based implementation code for the complex patterns [1]. We have initially applied model-to-text Java Emitter Template (JET) transformation to transform the UML activity model-based SOTA patterns into textual Java. JET is an open-source technology developed by IBM and are typically used in the implementation of a code generator [7]. Here, the transformation contains XPath expressions to navigate the UML activity models created for the patterns and extract model information dynamically to the transformation. However, as JET's support for UML models has several limitations, we have used a more effective method to transform the UML activity diagram-based SOTA patterns into textual Java.

This multi-stage transformation chain describes an effective pipeline of model-to-model and model-to-text JET transformations. In this solution, first a model-to-model mapping transformation was created which extracts relevant information from the UML activity model elements and stereotypes, and then a code generator specific EMF intermediate model was built which contains only information required for the back-end model-to-text JET transformation. The front-end model-to-model transformation automatically invokes the back-end JET transformation.

5.3. Platform-Specific Pattern Models. The transformations generated Java files for the SOTA patterns in e-mobility can be elaborated by the engineer to derive a complete implementation for the patterns. For this, the engineer can use as examples the set of domain-independent Java templates discussed in Sect. 3.3. In this study, we have instantiated these templates to implement the behaviour of the SCs and managers involved in the e-mobility case study scenario. To this end, we have implemented several self-adaptation scenarios. For example, (1) the e-vehicle's energy level is inadequate to follow the plan. This is managed by the AM class defined for state of charge on the vehicle SC class; (2) the e-vehicle's climate comfort level is not satisfied, which is handled by the AM class specified to handle climate comfort for the vehicle; (3) the driving style requirements (velocity, acceleration) are not satisfied by the e-vehicle for the user. This is managed by the AM class defined for driving style; (4) the e-vehicle has missed (or is going to miss) a deadline of a planned event. This is managed by the super AM class defined for routing; (5) the user preferences are not satisfied. This is handled by the super AM for user preferences; (6) the e-vehicle does not arrive at the booked time. This is handled by the AM for parking lot availability; (7) the e-vehicle leaves earlier/later than booked time. This is handled by the AM class defined for parking lot availability.

In this manner, we have implemented and validated the SOTA patterns in the e-mobility case study. This shows that the patterns can be simulated and implemented effectively and that they can be effectively applied to a case study.

5.3.1. SimSOTA Installation and Usage. The distribution scheme that will be adopted for SimSOTA relies on the Eclipse platform feature export. The entire SimSOTA tool can then be downloaded using the standard Eclipse update site mechanism. An update site is a mechanism for finding and installing features. In order to export the plug-ins in the SimSOTA tool, first, a feature project that references the plug-ins will be created. Second, an update site will be created to distribute the feature created. Finally, the Eclipse Update Manager can be used to scan update sites for the newly created feature for SimSOTA and install it. The installed plug-ins can be executed in the IBM Rational Software Architect simulation environment. At this moment, however, a packaged version of SimSOTA is not publicly available, due to its dependencies on the non-free IBM Rational Software Architect simulation environment.

6. Discussion and Related Work. Our approach and SimSOTA plug-in offers several benefits in the domain of self-adaptive architectural patterns engineering and implementation. SimSOTA is an integrated eclipse plug-in with several plug-ins grouped together, i.e. a simulation plug-in and transformation plug-ins. The idea is to provide an integrated environment for the architect to engineer and implement SOTA patterns in their respective domains. A key goal of SimSOTA simulation plug-in is to facilitate the engineering (modelling,

simulating and validating) of complex self-adaptive systems based on feedback loops. This is to facilitate general-purpose and application-independent instantiation of models for complex systems based on feedback loops. To achieve this, as described in the paper, we have provided a set of pattern templates (UML and Java) for all the key SOTA patterns in the catalogue. The SimSOTA plug-in can also be used to simulate and animate the patterns to better understand their complex and dynamic model behaviour. Also, the feedback loop models and their interplay can be validated to detect errors early and to check whether the specified behaviour works as intended. The transformation plug-ins of the integrated SimSOTA tool facilitate the automatic generation of implementation code in Java for the patterns. The use of transformations is beneficial here as it acts as a bridge to enforce correct separation of concerns between two design abstractions, i.e. UML-based patterns and their Java implementations. Other benefits include improving the quality and productivity of patterns development, and easing system maintenance and evolution for the patterns engineer.

Furthermore, our model-driven engineering approach integrates both decentralized and centralized feedback loop techniques in order to exploit their benefits. Centralized approaches allow global behaviour control, but they contain a single point of failure and suffer from scalability issues. Meanwhile, decentralized approaches do not require any a priori knowledge, nor do they contain a single point of failure. The integration of these approaches was evident with the use of a collection of architectural patterns at both SC (decentralized - e.g. `Autonomic SC pattern` and `Parallel AMs SC pattern`) and SCE (centralized - e.g. `Centralized SCE pattern`) levels.

There are several works in the literature that are related to the current research as presented next. Several authors (e.g. [12, 8, 15, 14, 16, 11, 20]) have emphasized the need to make feedback loops first-class entities in self-adaptive systems. Muller, Pezze and Shaw [12] discuss an approach to increase the visibility of control loops to support their continuous evolution. They highlight the need for multiple control loops in an adaptive ultra large-scale system, and stress the need for refining the loops into reference models and design patterns. Although these ideas have been discussed at the conceptual level, no implementation or validation of the work using techniques such as simulations has been reported [12].

Vromant et al. [15] describe an implementation framework that extends IBM's MAPE-K model with support for two types of adaptation coordination: intra-loop and inter-loop. While their work is comprehensive in coordinating and integrating multiple control loops, the MAPE-K loops used do not support an integration of centralized and decentralized adaptation coordination as provided in our work.

To manage the complexity of internet applications, the authors in [14] propose a set of weakly interacting (i.e. stigmergy, hierarchy and direct interaction) feedback structures where each structure consists of a set of feedback loops to maintain one system property. As in our work, in [16] a feedback loop has been represented as a flow of information and actions, and UML activity diagrams have been used as notation. However, unlike our approach, both [14] and [16] have not provided detailed individual steps of the adaptation process nor have they provided tool support for modelling, simulation and validation of the feedback loop models.

Vogel and Giese [20] establish a domain-specific modelling language for run-time models called *megamodels* and an interpreter to execute them. Using the modelling language, single and multiple feedback loops and their interplay can be explicitly specified in the megamodels. Like our approach, their work has considered detailed individual adaptation steps like monitoring, analysis, planning and execution. Also, their modelling language is similar to the UML activities used in our work with respect to modelling flows of actions or operations. However, the authors have not considered any validation of the simulated feedback loop models in [20].

Compared to previous approaches to the engineering and implementation of self-adaptive systems using feedback loops, to the best of our knowledge, there is very little implementation or tool support to address the needs of software architects. The present work aims to contribute to this end.

7. Conclusions and Future Work. In this paper, we presented SimSOTA, which is an integrated Eclipse plug-in tool we have developed to engineer and implement self-adaptive systems based on our feedback loop-based approach. SimSOTA facilitates both modelling and simulating of complex patterns, and the generation of Java-based implementation code for the patterns using transformations. The approach integrates both decentralized and centralized feedback loop techniques in order to exploit their associated benefits. The approach and plug-in have been validated and assessed using a case study in cooperative electric vehicles.

As part of our future work, we will exploit the results of our simulation and implementation experiences to produce effective software engineering guidelines to facilitate the development of self-adaptive application with

SimSOTA. Second, we plan to integrate the SOTA patterns defined at the conceptual, UML and Java levels with the rest of the e-mobility framework for further validation. Finally, we plan to assess SimSOTA in the context of different application areas, namely swarm robotics systems and distributed cloud systems.

Acknowledgments. This work is supported by the ASCENS project (EU FP7-FET, Contract No.257414).

REFERENCES

[1] D. B. Abeywickrama, N. Hoch, and F. Zambonelli, *An integrated Eclipse plug-in for engineering and implementing self-adaptive systems*, Proceedings of the IEEE 23rd International WETICE Conference, IEEE (2014), pp. 3–8.

[2] D. B. Abeywickrama, N. Bicocchi, and F. Zambonelli, *SOTA: Towards a general model for self-adaptive systems*, Proceedings of the IEEE 21st International WETICE Conference, IEEE (2012), pp. 48–53.

[3] D. B. Abeywickrama, and F. Zambonelli, *Model checking goal-oriented requirements for self-adaptive systems*, Proceedings of the IEEE 19th International Conference and Workshops on Engineering of Computer Based Systems (ECBS), IEEE (2012), pp. 33–42.

[4] D. B. Abeywickrama, N. Hoch, and F. Zambonelli, *SimSOTA: Engineering and simulating feedback loops for self-adaptive systems*, Proceedings of the 6th International C* Conference on Computer Science & Software Engineering (C3S2E'13), ACM (2013), pp. 139–144.

[5] D. B. Abeywickrama, F. Zambonelli, and N. Hoch, *Towards simulating architectural patterns for self-aware and self-adaptive systems*, Proceedings of the 2nd Awareness Workshop co-located with the SASO'12 Conference, IEEE (2012), pp. 133–138.

[6] G. Cabri, M. Puviani, and F. Zambonelli, *Towards a taxonomy of adaptive agent-based collaboration patterns for autonomic service ensembles*, Proceedings of the International Conference on Collaboration Technologies and Systems, IEEE (2011), pp. 508–515.

[7] J. DeCarlo, L. Ackerman, P. Elder, C. Busch, A. Lopez-Mancisidor, J. Kimura, and R. S. Balaji, Strategic Reuse with Asset-Based Development, IBM Corporation, Riverton, New Jersey, USA (2008).

[8] R. Hebig, H. Giese, and B. Becker, *Making control loops explicit when architecting self-adaptive systems*, Proceedings of the 2nd International Workshop on Self-Organizing Architectures, ACM (2010), pp. 21–28.

[9] N. Hoch, K. Zemmer, B. Werther, and R. Y. Siegwart, *Electric vehicle travel optimization—customer satisfaction despite resource constraints*, Proceedings of the 4th Intelligent Vehicles Symposium, IEEE (2012), pp. 172–177.

[10] J. O. Kephart and D. M. Chess, *The vision of autonomic computing*, IEEE Computer (2003), 36(1):41–50.

[11] M. Luckey, B. Nagel, C. Gerth, and G. Engels, *Adapt cases: extending use cases for adaptive systems*, Proceedings of the 6th International SEAMS Symposium, ACM (2011), pp. 30–39.

[12] H. Müller, M. Pezzè, and M. Shaw, *Visibility of control in adaptive systems*, Proceedings of the 2nd International Workshop on Ultra-large-scale Software-intensive Systems, ACM (2008), pp. 23–26.

[13] M. Puviani, G. Cabri, and F. Zambonelli, *A taxonomy of architectural patterns for self-adaptive systems*, Proceedings of the 6th International C* Conference on Computer Science & Software Engineering (C3S2E'13), ACM (2013), pp. 77–85.

[14] P. Van Roy, S. Haridi, and A. Reinefeld, *Designing robust and adaptive distributed systems with weakly interacting feedback structures*, Technical report, ICTEAM Institute, Universit catholique de Louvain (2011).

[15] P. Vromant, D. Weyns, S. Malek, and J. Andersson, *On interacting control loops in self-adaptive systems*, Proceedings of the 6th International SEAMS Symposium, ACM (2011), pp. 202–207.

[16] T. De Wolf and T. Holvoet, *Using UML 2 activity diagrams to design information flows and feedback-loops in self-organising emergent systems*, In T. De Wolf, F. Saffre, and R. Anthony, editors, Proceedings of the 2nd International Workshop on Engineering Emergence in Decentralised Autonomic Systems (2007), pp. 52–61.

[17] B. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee, et al, *Software engineering for self-adaptive systems: A research roadmap*, Software Engineering for Self-Adaptive Systems, volume 5525 of *LNCS*, Springer-Verlag (2009), pp. 1–26.

[18] E. Clayberg and D. Rubel, *Eclipse Plug-ins*, Addison-Wesley Professional, 3 edition, 2008.

[19] T. Haupt, *Towards mediation-based self-healing of data-driven business processes*, Proceedings of the 7th International SEAMS Symposium, IEEE/ACM (2012), pp. 139–144.

[20] T. Vogel and H. Giese, *A language for feedback loops in self-adaptive systems: Executable runtime megamodels*, Proceedings of the 7th International SEAMS Symposium, IEEE/ACM (2012), pp. 129–138.

[21] J. Wuttke, Y. Brun, A. Gorla, and J. Ramaswamy, *Traffic routing for evaluating self-adaptation*, Proceedings of the 7th International SEAMS Symposium, IEEE/ACM (2012), pp. 27–32.

Edited by: Giacomo Cabri
Received: September 15, 2014
Accepted: January 5, 2015

DOI 10.12694/scpe.v15i4.1053
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 15, Number 4, pp. 309–329. http://www.scpe.org

SIMULATION DATA SHARING TO FOSTER TEAMWORK COLLABORATION

CLAUDIO GARGIULO* DELFINA MALANDRINO† DONATO PIROZZI‡ AND VITTORIO SCARANO§

Abstract.
This paper introduces Floasys, a Web-based platform to foster the collaboration among engineers involved in Computational Fluid Dynamics (CFD) simulations. The platform has been designed around the simulation data, i.e., fostering and stimulating sharing, re-use and aggregation of models, simulation results and engineers annotations. Floasys requirements come directly from an extensive requirements study that we conducted with two different teams in Automobiles (FCA), geographically distributed, who daily perform an intense activity of CFD simulations to design vehicle products. Collaborative requirements were gathered through stakeholders' interviews and a user survey. We describe, first, Functional and Non-Functional requirements as suggested by relevant literature (both in scientific and industrial setting) and by the user survey performed within FCA teams. Then, we show Floasys functionalities and its architecture, that is based on a centrally managed repository of simulation data. By enriching the repository with metadata annotations, Floasys provides all the desired functionalities to allow CFD analysts an easy and immediate access to simulation data and results performed within the teams so that they can leverage them to make the right design decisions.

In this paper, we were able to (1) identify key collaborative requirements for CFD design, (2) address each of them with an integrated, extensible and modular architecture, (3) implement a working industrial prototype (currently under testing and evaluation in a real setting like FCA), and (4) identify the possible extensions to different contexts (like aeronautic, rail and naval sectors).

Key words: Data sharing, model sharing, collaboration, simulation survey, CFD simulators integration, Web-based simulation, simulation tagging, simulation search, simulation data version control.

AMS subject classifications. 68U20

1. Introduction.

Nowadays, SMEs and large industries extensively use simulations to design new products. Products became even more complex, they integrate many components (e.g., more than 20000 separate components for automotive product [1]) and are available to customers in many configurations. To manage this complexity, to get "*a better insight into product behaviour*" [2], and to reduce costs for prototypes [3], industries use different types of computer simulations [3] to simulate and analyse the products behaviour. One type of simulation is Computational Fluid Dynamics (CFD) used to investigate the physical product behaviour, such as external aerodynamics, underhood cooling, air conditioning and so on. In addition, SMEs and large industries have many locations, therefore, both co-located and geographically distributed engineers need to collaborate together, share simulation models, know-how, best practices and other important information.

This paper introduces Floasys, a Web-based platform designed to support simulation data, knowledge and result sharing among CFD analysts in Fiat Chrysler Automobiles (FCA). The goal is **to promote the sharing of simulation models and results to foster their reuse** among engineers. This work introduces, analyses and discusses Functional and Non-Functional collaborative requirements (Section 2) as suggested by relevant literature (both in scientific and industrial setting) and by results of an extensive user survey performed within FCA teams. The collaborative requirements are: *simulation data centralisation, metadata over simulation data, search facility, version control over data and data sharing.* Functional and Non-Functional requirements led the design of Floasys' architecture and its functionalities. Floasys collects and centralises simulation data over time. Simulation data are collected from multiple simulators and are stored in open format (e.g., XML). Floasys provides additional services over collected simulation data. It provides a Search tool that is independent by the specific simulator. It is very useful to get simulations performed by different engineers to compare performance about multiple design revisions. In addition, it allows the data sharing through URLs exchange. The Floasys target customers are industries who use CFD simulators to design their products. From architectural point of view, Floasys meets the extensibility and modularity Non-Functional requirements since it can be tailored to customer needs, accommodate future needs and used in many departments. Although this work concerns an automotive use case, issues that we are facing within this sector seems to be very common issues also in other

*R&D - Aerothermal CFD, FIAT Chrysler Automobiles, Italy, claudio.gargiulo@fiat.com
†ISISLab, Dipartimento di Informatica, Università di Salerno, Italy, delmal@dia.unisa.it
‡ISISLab, Dipartimento di Informatica, Università di Salerno, Italy, dpirozzi@unisa.it
§ISISLab, Dipartimento di Informatica, Università di Salerno, Italy, vitsca@dia.unisa.it

sectors as highlighted in [2, 4], especially for the list of gathered requirements. Therefore, we believe that many of our considerations and design decisions could be adopted also for other type of simulations and in other contexts (i.e., aeronautic, rail and naval sectors).

The paper is organized as follows. Section 2 briefly introduces the use case study and outlines who are the stakeholders. The aim is to provide an overview about the context and its internal organization. Then, it analyses the collaborative Functional and Non-Functional requirements. Section 3 introduces the Floasys prototype with its functionalities. Section 4 discusses the Floasys architecture design decisions to meet stakeholders' requirements. Through the paper, we track and map the collaborative requirements with the solution ideas and the specific implementation technologies (i.e., libraries) used to develop the Floasys architecture. Section 6 concludes the paper and discusses possible future works.

2. Collaborative Requirements. This section analyses the key collaborative Functional and Non-Functional requirements to design a platform to foster the collaboration among industrial simulation practitioners and promote the sharing of models, results, and know-how. These requirements come from a relevant literature study and an extensive requirements elicitation activity performed through observations, stakeholders interviews and a user survey (Appendix A).

Fiat Chrysler Automobiles (FCA), as many other large industries, is organised in multiple geographically distributed teams that collaborate together. Through our survey analysis, we get that all analysts collaborate at least with another engineer in the same office and more than half analysts collaborate with at least one engineer who works in another location. They collaborate together sharing file geometries (CAD files), simulations and documents (e.g., slides, spreadsheets).

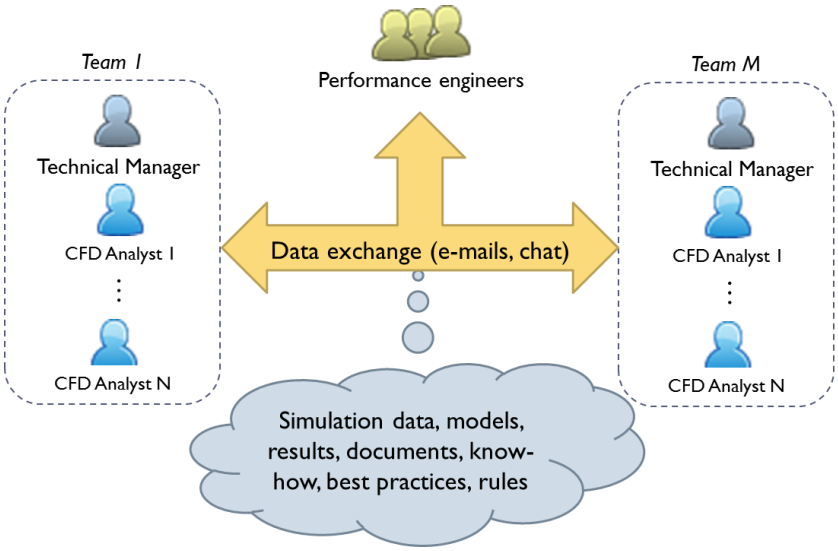

FIG. 2.1. *Geographically distributed teams that collaborate together in asynchronous way.*

Large industries have multiple locations around the world and are internally organized in multiple structures of different types. One type of structure is the *functional area*. Functional areas have technical know-how about a specific sector (i.e., engineering, cost engineering, marketing, commercial). Specifically, engineering functional areas perform tasks to design products and constantly invest in Research and Development (R&D) to improve their know-how and to be ready to provide innovative design solutions. The Computational Fluid Dynamics (CFD) unit is the engineering functional area with highly skilled engineers, called CFD analysts, who perform numerical computer simulations to analyse problems that involve fluid flow and other related physical phenomena, such as aerodynamic, aerothermal and aeroacoustic automotive product behaviour. CFD is widely adopted in many industrial sectors, such as automotive, aerospace, high-tech and chemical sectors. CFD analysts perform simulations following the CFD Workflow [5] that is iterative and consists of three phases: (1) pre-processing to prepare simulation, (2) solving and (3) post-processing to analyse results. The CFD unit

and the CFD Workflow are our use cases. In each CFD unit, there are analysts and a technical manager who is responsible for the internal team organisation, resources monitoring and their allocations.

In a large industry, many CFD units collaborate together (Fig. 2.1). The collaboration is among geographically distributed CFD units and, among CFD units and other industrial teams, such as the product style designers and the performance engineers. In order to design an automotive product many engineers collaborate together. Especially, to perform aerodynamics/aerothermal analyses, CFD analysts, automotive designers, and performance engineers collaborate together.

The prerequisite to enable the collaboration among analysts is the ***simulation data centralisation***. Industries perform many simulations per year, therefore, in order to foster the ***model reuse*** and promote the ***data sharing***, it is fundamental how easy it is to retrieve the needed data stored in multiple repositories with different formats (often in closed file format). In order to improve data retrieval, users aim to annotate simulation files with additional ***metadata over data***, such as free tags or structured data, and to have a ***search tool*** able to get desired data. Search tools should support at least the search through files' names, annotated metadata and simulations' contents. ***Simulation data version control*** is another desired feature. The aim is to have a history of modifications made to simulations. It is a desired feature because the same simulation is often performed changing only some parameters (e.g., inlet velocity).

TABLE 2.1
Stakeholders' Collaborative Requirements.

	Requirement	**Notes**
Req. 1	*Simulation Data centralisation*	
Req. 2	*Metadata over simulation data*	Link metadata to simulations (e.g., free tags).
Req. 3	*Search facility*	Search based on file names, file content and tags.
Req. 4	*Version control over data*	
Req. 5	*Data sharing*	
Req. 6	*Integrate multiple simulators*	Avoid Vendor Lock-In
Req. 7	*Extensibility and modularity*	
Req. 8	*Do not change how engineers work*	

In order to gather the collaborative requirements (Table 2.1), we worked closely with a team of professionals in Pomigliano D'Arco (Italy) who extensively use CFD simulations to design automotive products. We observed their daily work annotating, collecting and analysing their tasks and workflows. We constantly discussed with analysts and technical managers trying to get a deep understanding of their work and answer to our questions. Requirements are refined through continuous iterations. FCA has multiple geographically distributed teams, therefore in order to get the collaborative requirements directly from stakeholders, we issued an electronic survey (shown in Appendix A) created with Google Forms[1]. The survey questions were divided in the following main sections: *participants' experience, collaboration among engineers and data sharing, data centralisation and data search*, and *simulation data versioning*. The survey responders are seventeen FCA professionals half from Pomigliano D'Arco (Naples, Italy) and half from Orbassano (Turin, Italy). Both groups design products using Computational Fluid Dynamics simulations. Through the paper we sometimes differentiate the technical managers and the analysts because they have different roles and requirements. Technical managers usually ask management features, such as the opportunity to monitor resources, projects timeline and performance goals. On the other hand, CFD analysts, who perform simulations, require engineering features (e.g., simulation monitoring, automatic document generation). Of course, both roles aim to collaborate over centralised data at different granularity. Floasys has been designed to also support engineering tasks, such as the simulation convergence monitoring, engineering wizards to automate repetitive tasks, simulation templates and so on. In this paper we mainly focus on the collaborative aspects overlooking the engineering Floasys's features. An important consideration is the impossibility to change how the employers actually work. Any architectural software solution to meet the requirements shown in Table 2.1 must rely on existing internal procedures and must not change them. During the requirement elicitation activity we also tried to understand the ways on how a collaborative platform could be introduced and deployed over existing practices ***without hardly change***

[1] http://www.google.com/google-d-s/createforms.html

how the engineers work but at same time improving their work. The following section will analyse each requirement listed in Table 2.1.

2.1. Simulation Data Centralisation. In order to support collaboration among engineers (Fig. 2.1) they must access to centrally available simulation data (Req. 1, Table 2.1). The idea is to collect data from different sources over time (i.e., from different simulators) and store them in open format like XML. In this way, data and results can be aggregated in different ways and can be compared within the same project or among different projects. Performance engineers and technical managers need to work on aggregate data (e.g., statistical data, trends about performances) whereas CFD analysts access to fine grain simulation data (e.g., model, simulation case) and their results to perform comparison. Obviously, data aggregation is not feasible with classic shared network folders that store data in a closed file format. Actually it is manually performed with continuous copy-and-paste operations among simulators and documents. In according to Aberdeen Group's whitepaper "*Getting Product Right the First Time with CFD*" [2], in order to improve the company competitiveness, they should centralise simulations results. Our aim is to centralise simulations and all their related data, such as the 3D geometries, simulation setup parameters and documents supporting their retrieval. In order to centralise data and provide additional services over them, software designers should consider: file size, total number of performed simulations and closed file format. In our use case, both geometries and simulations are very large files. In the survey, we asked which are usually the geometries and the simulations file sizes (questions **Q5** and **Q6** of the Survey shown in Appendix A). Figure 2.2a shows that the CAD file size is about one gigabyte in the fifty percent of answers. The file geometry can also contain the surface mesh and/or the volume mesh, explaining the differences of file size answers depicted in the chart of Figure 2.2a. Instead, the simulation file size (Fig. 2.2b) is more than ten gigabyte in 80% of answers. Simulations are so large because they contain the entire detailed vehicle geometry, the surface and the volume mesh as well as the physical/mathematical data to describe the model.

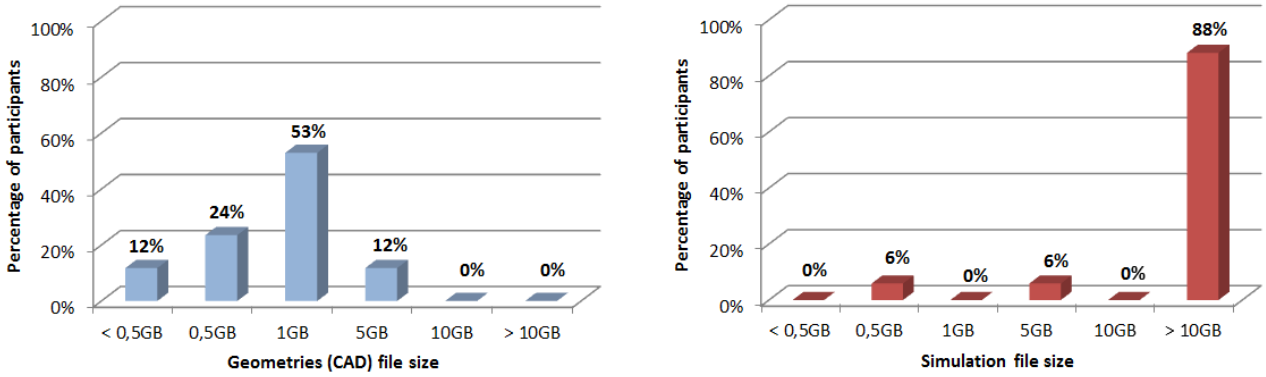

FIG. 2.2. *Geometries (question Q7) and simulations file size (question Q8).*

An alternative idea to provide services like data search or results aggregation, is to use a rational database to store simulation data, but considering file sizes and huge number of simulations we excluded it. In order to solve simulations the original files can not be moved and must be stored in their original format on file system. The use of a database leads to continuous transfers of data from the database to the file system and vice versa, compromising performance and response time.

2.2. Provide Search Facility. The aim is to provide a search tool able to find data using simulation file names, simulation content (e.g., its model, parameters, etc.) and metadata (e.g., tags). Simulators software often store simulation data as binary files in a closed file format. In addition, the used CFD simulator does not have an export functionality to an open format. Therefore, classical search tools are not useful to find simulation files based on their content (files are in binary format). For instance, the Windows OS search utility can not be used to search within the file content. To overcome this issue, users actually insert a lot of information in the simulation file name that will be useful to find data the next time. As shown in Figure 2.3, the main

information inserted in the file name are (questions Q13-Q18): the project, the release, the revision number, the engine model and the vehicle trimming. Users decide to put the most important information, regarding their personal opinion, in the file name with the drawback to have very long file names. In addition, not all information can be stored in the file name so a lot of data remain within the simulation closed file and can not be used for next retrievals.

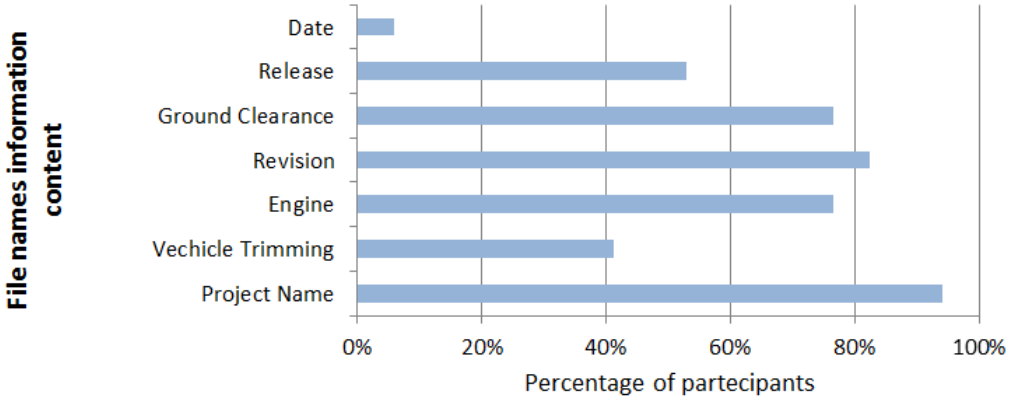

FIG. 2.3. *Information inserted in the simulation file names (multiple choices question Q12).*

More than half of analysts follow roughly some rules to store files in shared file system trying to follow them over time. Here, the term "rules" mainly means how engineers give a name to a file and how they decide the directories structures to improve the future simulation retrieval. Nevertheless these rules are mostly a personal choice (82%), engineers add essentially the same information to file names because the analysed engineering field is very specific. The limitation of this approach emerges when an engineer needs to search a simulation performed by other employees, mostly because he can not use existing search tools (e.g., the Windows Search tool) to search simulations based on the file content. An example of query is: "*search all simulations performed at inlet velocity X [km/h] that has the spoiler*". Unfortunately these data are not inserted in the file name and remain inside the closed files. This also limits the aggregation of data at different levels based on specific keywords and the relative results comparison of multiple different simulations to generate performance history charts.

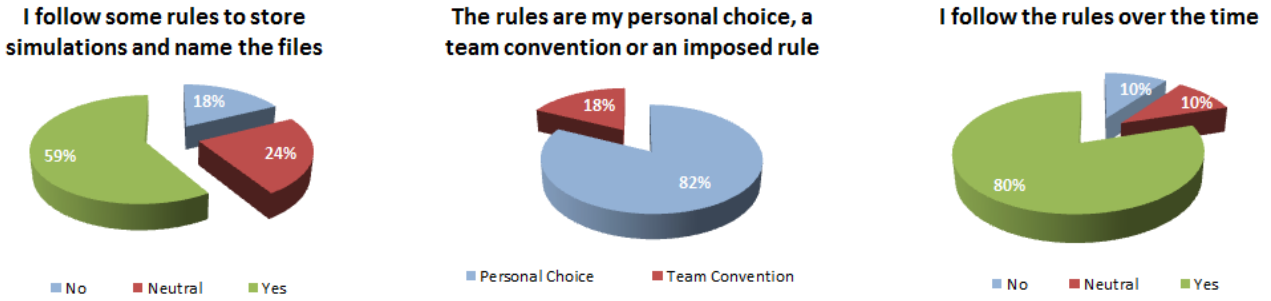

FIG. 2.4. *Rules to store the files on shared network folder (questions Q11, Q13, Q14).*

2.3. Provide Metadata over Simulation Data. Engineers use multiple simulators software, some of them store data in closed file format. As stated in the previous section, the file content can not be used to retrieve the files using the classical search tools such as using the Operating System find tool. Actually, to overcome this issue, engineers insert a lot of information in the simulation file name such as project name, revision and engine type (Fig. 2.3). Obviously, the file name can not host too many data, so other useful

data are not annotated with simulations (e.g., free comments, descriptions). To get this requirement through interviews and the survey we asked whether the engineers desire to link other data to files (question Q15). All analysts (100%) desire a system to link other information to the files, such as the file tagging.

2.4. Simulation Data Versioning. As reported in the Aberdeen Group market research [2], an action to improve the company competitiveness is to provide version control over data. Our survey aims to further investigate this need especially to understand its value for stakeholders. Version control means that users can track modifications made to a simulation over time. It is interesting because engineers usually do not start simulations from scratch but they copy an existing file changing some parameters. In addition, starting from the same simulation file many other simulations can be performed just changing few parameters (e.g., the inlet velocity). In according to our survey, more than 60% of participants declared that they do not have a tool to track the simulations modifications. In addition more than 80% of participants said that the feature could be useful.

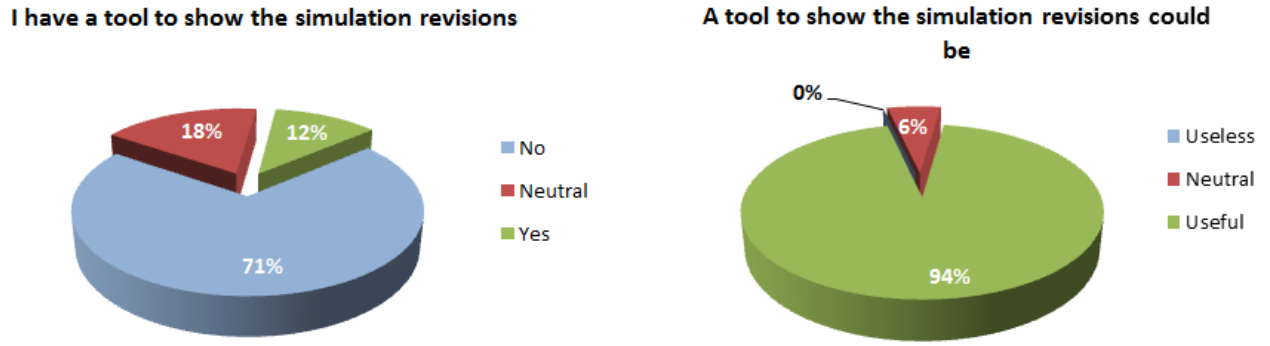

FIG. 2.5. *Version control (question Q20).*

2.5. Support Data Sharing. CFD analysts need a mechanism to exchange references about data. On Internet a common way to share resources is exchanging URLs. Hence, our idea is to univocally identify simulation data with URLs and use them to share data among engineers. An important aspect of this technique is *"who can see what data"*. Multiple industrial roles exists (Fig. 2.1), so an access control is important to control the sharing of confidential data.

2.6. Simulator Independence and Integration of Multiple CFD Simulators. The previous requirements must work independently by specific used simulators to generate data. For instance, tagging and search functions must work on a repository of heterogeneous simulations coming from multiple simulators. This requirement is very important because in the analysed context, analysts use multiple CFD software and actually one single software can not be used to perform all simulation types. In our use case and large industries, there are different teams that use different software to perform tasks. For instance, a team is responsible for the CAD design whereas another team simulates models using other software. Obviously, in other contexts both design and simulations can be done by the same team with an all-in-one CAD/CAE software. Through the survey, we asked (multiple choices question Q21) to indicate which simulator software the analysts use, to give an idea about their multiplicity. All analysts use Star-CCM+ and more than half of them use OpenFOAM. Other used software are: CFD++ (35%) and PowerFlow (18%). Analysts have used software over the years and they are confident with them. Moreover, industries are unwilling to invest in training engineers on other software products. Therefore, in order to meet the requirements is fundamental to support and collect data from multiple daily used CFD simulators. This is a key difference with other platforms (i.e., e-Science) that often integrate simplified or in-house developed solvers [6].

It is evident that any platform must consider the integration of multiple simulators. The integration of multiple simulators (Req. 6 in Table 2.1) has some difficulties especially because CFD analysts use often proprietary software and actually a lack of simulator standardisation exists so that many software do not have

function to export data in open format. The import/export in open format are important functions to evaluate during the choose of a CAD/CAE software [7] otherwise simulation data are locked in the vendor software. Vendor Lock-In is a well-known Anti-Pattern [8] [9] [10]: the phenomenon that causes customer dependency on given vendor about a specific good or service [11] with high switching costs [12]. Vendor Lock-In occurs both in terms of services and data. Vendor Lock-In Anti-Pattern in terms of services occurs when the architecture heavily relies on a closed vendor software and strictly depends on vendor choices, so the architecture is product-dependent [13]. Data Lock-In occurs when the only way to access to data is using the Vendor Software because data are stored in a proprietary file format or on a vendor server that does not provide an export functionality to open format or a public customer API. The exporting and importing of geometric data are well-established functionalities for CFD software, simply because they must commercially support the interaction with other CAD software. Conversely, it is not the same for simulation data such as case setup, simulation results and so on. Data Lock-In is very common in Cloud Environments [14] and is an obstacle to cloud computing [15]. Vendors lock users in to make difficult to change product because they cannot get their data; despite, as reported in literature, giving the opportunity for customers to get their data increases their trust in the product [16]. A design solution useful to mitigate the Vendor Lock-In is to design the system with an additional layer called isolation layer [8].

2.7. Extensibility and Modularity. The combination of modularity and extensibility [17, 18] system qualities advantages are: the opportunity to compose a system with the only needed modules, the introduction of new functionalities tailored to customers' needs, and the creation of customers own modules to automatise specific tasks keeping them private to protect the know-how. Extensibility is the ability of a software system to allow and accept significant extension of its capabilities without major rewriting of code [17] [18]. Extensibility is a quality architecture attribute useful during the development and especially in future when more and more simulators' features will be integrated in the architecture [19]. Industries want to deploy the same system with different features. Modularity "*is the degree to which a system or computer program is composed by discrete components such that a change to one component has minimal impact on other components*" [17]. The architecture must be modular to support both the adding of new simulators and the removing of existent simulators. The modularity requirement has an interesting advantage for the architecture design: the engineering tools and simulators are loosely coupled. An important consideration concerns the software license. Two opposite needs must be taken into account: on one hand, industries want to protect their know-how, on the other hand, the architecture must be also adopted in other contexts. Based on our use case, modularity, extensibility and EPL license [20] are the right mix. The architecture, the framework and some other modules are open source. At same time industries can protect their know-how developing their own private and closed modules.

3. Floasys Functionalities. This section describes the Floasys functionalities and shows its graphical user interface (GUI). Floasys provides a *simulator independent repository tool* to navigate open format simulation data repositories and annotate selected files through free and structured tags (Req. 2). Floasys has a structured and assisted *Search tool* to get simulations performed by different engineers (Req. 3) and *share* them (Req. 5). Floasys's screenshots contain CFD related data but its GUI and its ideas are general to be reused in other engineering areas (e.g., ergonomics).

Floasys is a Web-based platform to support both engineering tasks (e.g., run simulation, monitor simulations, generate documentation automatically etc.) and data sharing among dispersed engineers. Floasys centralises simulation data in open format and provides a search tool able to browse and query the simulation database using tags identifying versions, interesting features and open comments. The Figure 3.1 depicts a real-world Floasys workflow that is difficult or time-consuming without our platform. It is composed by six tasks executed in sequence. In Task 1, user finds a simulation using keywords like project name, revision, velocity and so on. The velocity is an internal simulation parameter. It is embedded in the closed file format, so the task to search by velocity can not be accomplished without Floasys or at least, as come to light in Section 2, the user can remember where he stored the simulation file and open it to check the velocity value. In addition, Operating System find tool can not be used to get the simulation because velocity is not included in the simulation file name (Fig. 3.2). With Tasks 2 and 3, the user selects a simulation from the list of results to get the original simulation file and open it with the proprietary software. Unfortunately, the original simulation file is not in the repository. Using Floasys, nevertheless the original file was deleted, the user can get the simulation data, setup

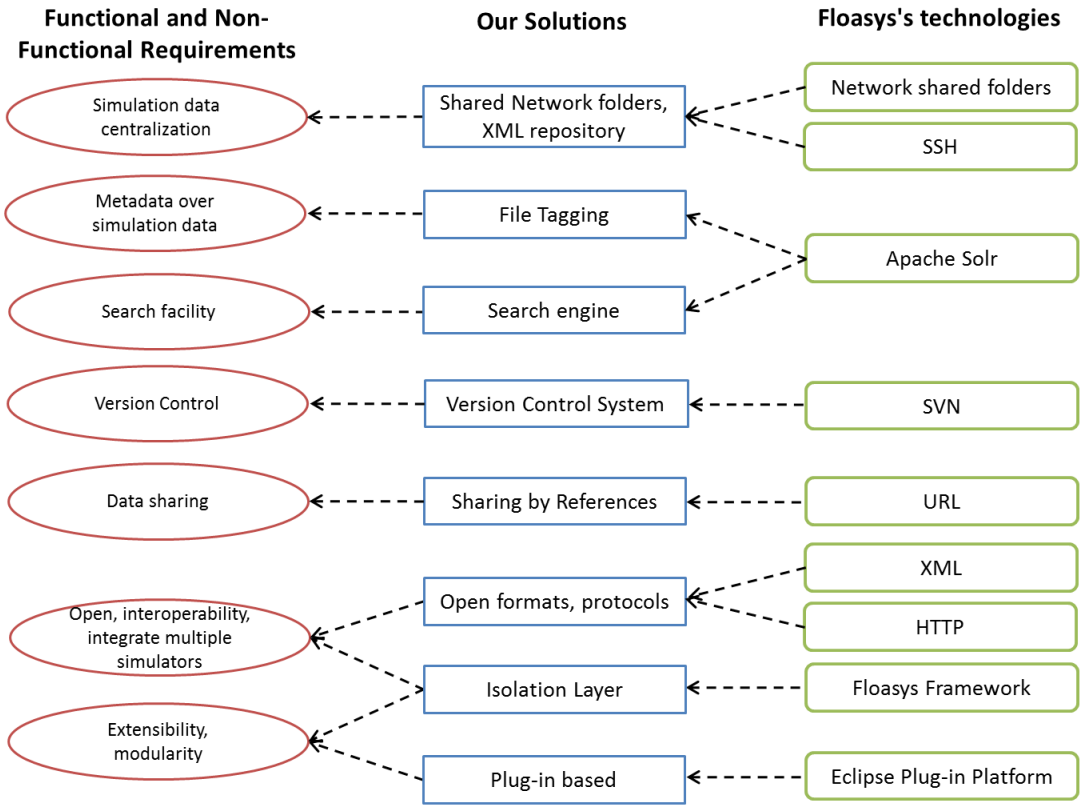

FIG. 2.6. *Mapping among Stakeholders' requirements, solutions and prototype technologies.*

and results. Of course, these data can not be used directly to simulate it again. Anyway, an expert engineer can recreate the simulation starting from the provided surface mesh and simulation setup (boundary conditions, physical model, used parameters, previous reports and so on). The Task 6 concerns the sharing of a simulation URL to another user via a preferred medium (e.g., e-mail, chat). Of course, the shared URL is available only within the industry's Intranet.

Floasys provides a re-configurable GUI based on Perspectives and Views concepts provided by Eclipse Remote Application Platform (RAP) [21]. The idea is that the virtual workbench changes according to the engineering tasks. In this way, the system is able to show only relevant functionalities to perform the actual task. A *perspective* is a specific configuration of the workbench and contains many views to show information. A perspective provides well-organized software functionalities access because it divides them in semantically homogeneous sections.

3.1. System Independent Repository Tool and Simulations Tagging. The Repository tool supports the navigation of central simulation repositories. Floasys integrates multiple simulators, so data heterogeneity is one of the issues to face. For instance, OpenFOAM stores data in a well-defined directories structure of three folders (e.g., system, constant and iteration directories) and data are stored in multiple files. Instead, Star-CCM+ stores all simulation data in one single-vendor format file. OpenFOAM files are plain-text readable without the software, instead Star-CCM+ files are in closed format and they can be read only using the vendor software. The Repository tool, relaying on Floasys framework services, is simulator independent and is able to manage data from different simulators. The Repository tool inherits the user file system access permissions, so logged user can access only to files he/she has authorised. Floasys can access to network folder through a server using a SSH connection with logged user credentials.

The Repository tool provides file annotation and tagging features. The idea is to enrich simulations files

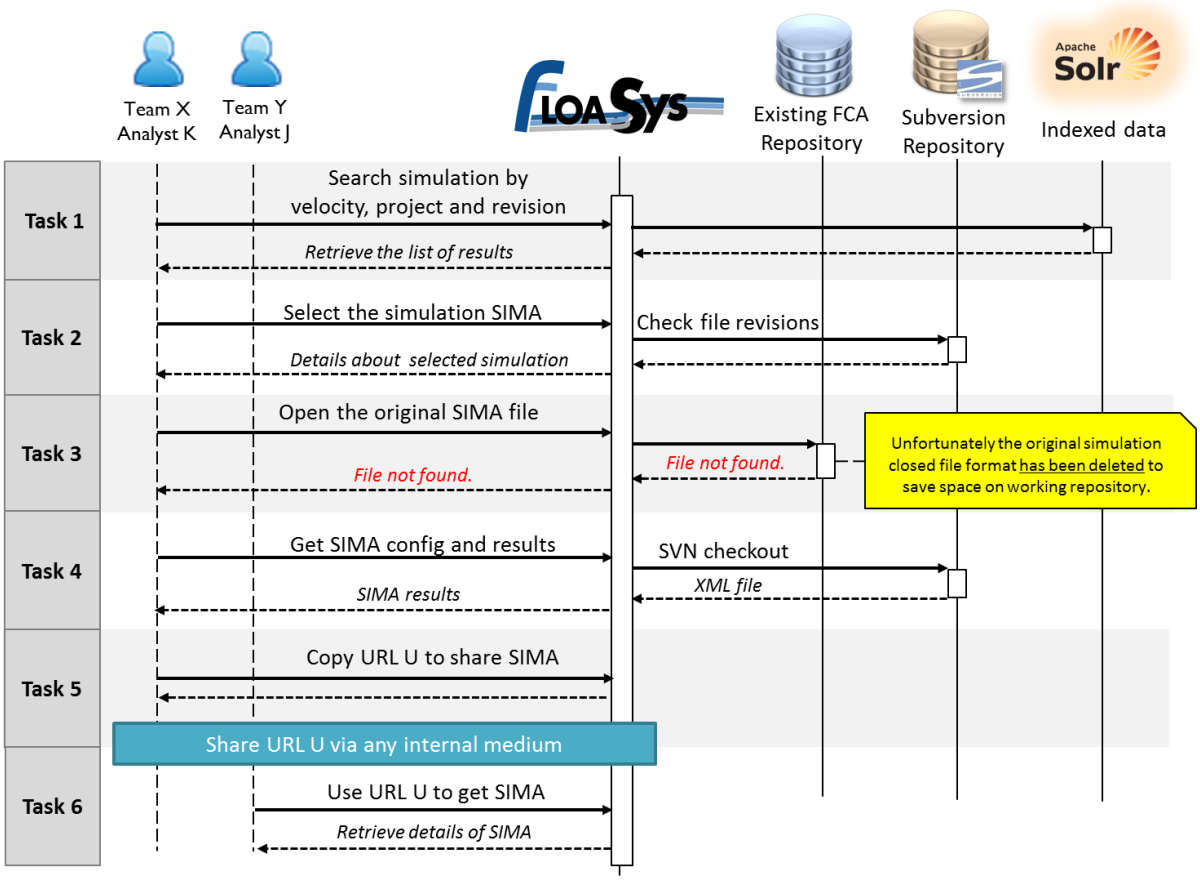

FIG. 3.1. *Example of a typical workflow supported by Floasys.*

with metadata: a user can annotate a simulation file and provide additional information useful to retrieve and share it in future. Examples of free tag categories are: brand, project name, revision and engine type; all information that can not be stored directly within simulation files, whereas Floasys allows it. Analysts are free to add any tag to files. In order to uniform the provided tags, during typing, Floasys suggests the tags to use (Fig. 3.2). Tags are both unstructured with free tags and structured inserted filling out standard forms like in Figure 3.3.

3.2. Search Tool and Data sharing. The Search tool (Fig. 3.4) is a Floasys perspective developed to provide the search of simulation data stored in central repositories. The tool supports the search by file name, simulation content, free tags and structured data (Req. 3 in Table 2.1). When a user types the search keywords, Floasys recommends further keywords to refine the search (Fig. 3.4). In this way, the tool supports the search activity suggesting further search keys to reduce the total number of potential results. The system performs search using only indexed data without accessing (e.g., open) to original closed format files. The results are displayed in a list. In order to display the revisions history, the user can select a simulation from the list of results.

Each simulation file has a unique ID within Floasys and all relevant data (e.g., documents, simplified 3D geometry, surface mesh and so on) are linked to this ID. Both repository and search tools provide a unique URL for each selected simulation. Our idea is to share data by simply exchanging unique reference to the specific simulation data. URLs identify simulation data and inherit file system permissions. The URL is private and is accessible only within the industry boundaries. Considering the Computer Supported Cooperative Work

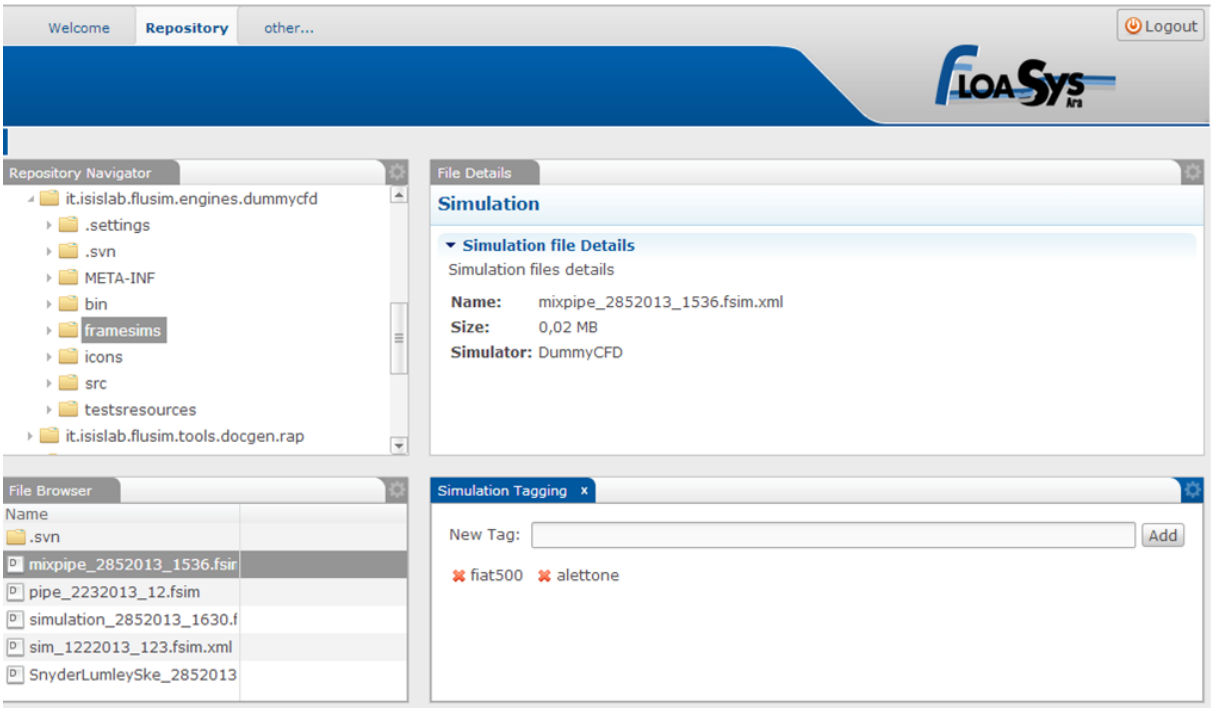

FIG. 3.2. *Repository tool to navigate a simulation repository and tag the resources (e.g., files).*

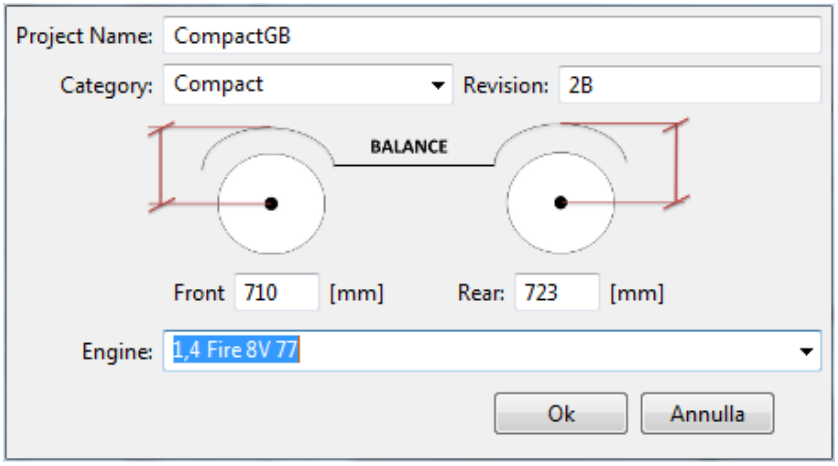

FIG. 3.3. *An example of structured data to store.*

(CSCW) space-time quadrants [22], Floasys supports the *asynchronous* data sharing for both co-located and distributed teams.

3.3. Web-based 3D Model Visualisation. Floasys shows a reduced 3D geometry of the simulated vehicle. Through this tool, engineers can quicly discover which components have been used to simulate the product without opening the CAD software. The tool shows a list of components with their Property IDs (PID) on the left (Fig. 3.5). The user can activate or deactivate some parts and can perform the basic zoom and pan operations. Figure 3.5 shows the simplified 3D surface geometry of a FCA production vehicle. The 3D vehicle geometries usually are very complex. To give an idea, each geometric model takes up ten gigabytes and engineers

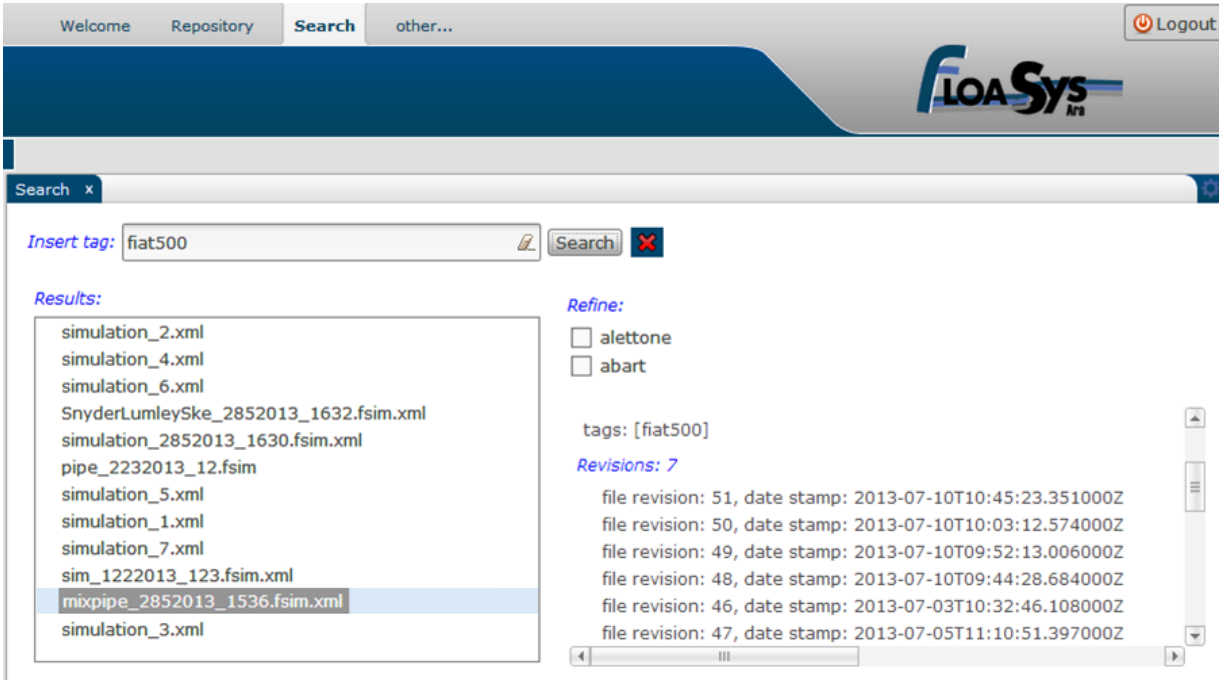

FIG. 3.4. *Search Tool*

use very performing hardware to open and manipulate them. An important requirement for any engineering platform is the visualisation of 3D geometric data. As many other platforms, Floasys is a Web-based platform. The vehicle geometries are impossible to render in the browsers using WebGL because they are very detailed and heavy; also the quantity of data to transfer from the server to the clients is very huge. To overcome this common issue and considering that the geometric representation is useful to give an immediate feedback on which components are included in the simulation, Floasys generates a simplified geometry representation to be rendered in the browser. Engineers need to have numerical tabular data, contour-plots and the 3D geometric model in the same view. Floasys provides a reduced geometry visualisation allowing engineers to quickly check which are the vehicle components at a glance. For instance, an engineer can visually check if the vehicle is simulated with the spoiler.

4. Floasys Architecture. This section introduces the Floasys architectural solution to centralise, annotate, tag, search and share simulation data. In order to meet the stakeholders' requirements, our solution collects simulation data from already existing simulation repositories (e.g., network shared folders), transforms, indexes (to provide high data retrieval performance) and store them in open format (e.g., XML).

4.1. Architecture Overview. Floasys is based on a Client/Server architecture (Fig. 4.1) developed using Eclipse Remote Application Platform (RAP) [21]. Clients are Web-based components. Therefore, Floasys is accessible through any browser installed on the company workstations. The Web-Based RAP clients communicate with the server exchanging commands and messages in JSON text format [23] over the HTTP protocol. Servers tend to interact with user browsers using the JSON exchange format [24] because it is easily parsed in client-side JavaScript language [23]. The Floasys's server can access to a set of already existing repositories (mainly shared network folders) that store the simulation files in their original format. In according to the internal policies, Floasys accesses to these existing FCA repositories in a read-only mode through the SSH protocol with the logged user credentials. Therefore, the architecture needs an additional repository to store simulations in open format (e.g., XML) with annotations, tags and additional metadata (Req. 2). Floasys supports two types of repository: an internal Subversion server or a shared network folder (without the version control support). In order to improve retrieval performances, Floasys indexes open format XML documents

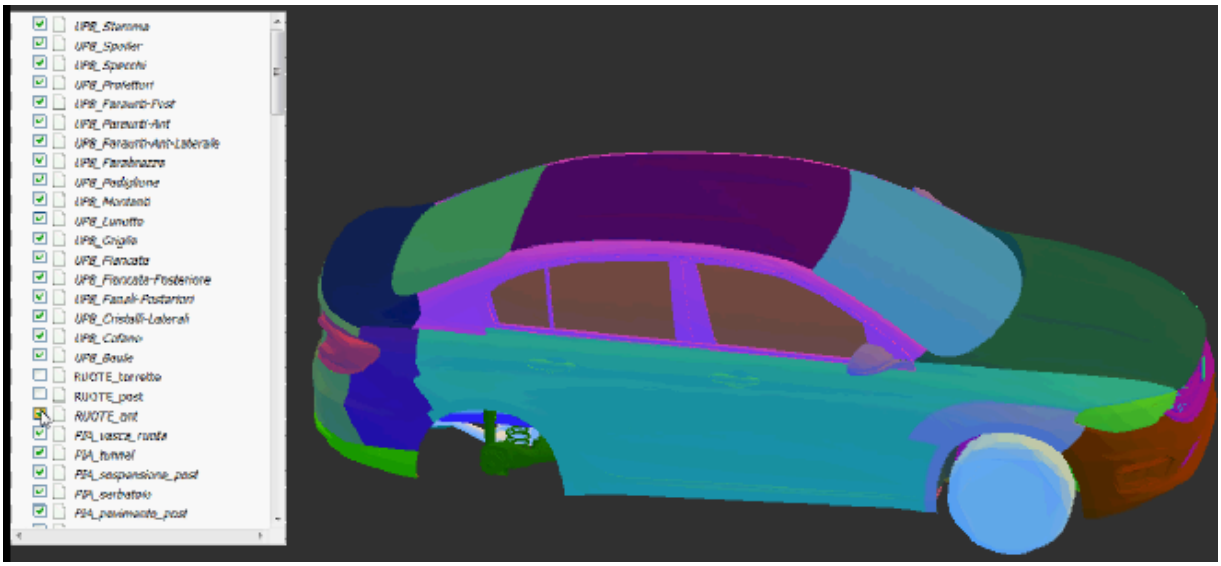

FIG. 3.5. *Floasys 3D model visualisation.*

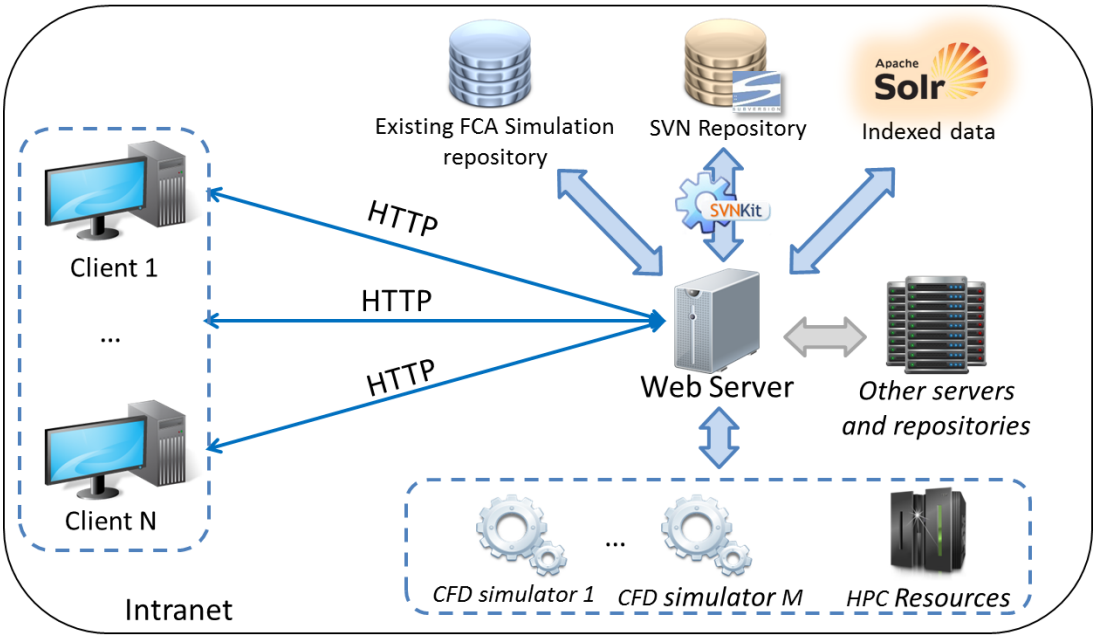

FIG. 4.1. *Floasys Client/Server Architecture.*

relaying on a well-established search engine technology like Apache Solr [25, 26, 27]. The server can access also to simulator software and High Performance Computing (HPC) resources as well as other internal services like the authentication service. Floasys is Intranet-based for security reasons. In addition, any kind of control access to data must be compliant with the industries internal policies and can not be override. To provide authentication and to manage both users and groups, Floasys can rely on existing industrial internal Lightweight Directory Access Protocol (LDAP) servers [28, 29] or use existing Secure SHell (SSH) accounts comply with existing file and directories access permissions. Floasys could be exposed also on Internet, but limitations exist

such as the huge amount of simulation data (gigabytes) to transfer. Trusting and security issues must be taken into account (e.g., to avoid espionage activities). Floasys is designed, developed and tested following an Agile methodology based on short iterations of two weeks each in average, delivering small functionalities every time. During the development, especially for server-side features, we wrote black box unit tests using JUnit [30]. From functionalities testing point of view, for each planned release we had a test plan with the test cases to execute and check on a controlled environment software installation. In addition, during the Floasys development, we worked closely to analysts in Fiat Chrysler Automobiles to get the user feedback as soon as possible that were recorded in an issue tracking system (e.g., Edgewall Software Trac[2]) and scheduled for the next plans in according to the issue/enhancement priority.

4.2. Server-side Software Architecture. The Floasys server-side component interacts with the simulator software to collect closed format data and transform them in open format. The architecture is a three layers approach (Fig. 4.2). It integrates multiple simulators in the bottom layer wrapping the vendor software. The top layer is the front-end that contains the Web-based GUI tools (or applications). The middle layer (1) provides a common APIs to the front-end tools, (2) provides a common unified data representation called Simulation Model for data coming from different vendor systems, (3) it is an isolation layer [8] to decouple the front-end from vendor-specific simulator wrappers and, (4) it allows the vendor-product switching at run-time to choose which ones are able to provide the needed services and data.

The middle isolation layer contains the common APIs exposed to the upper applications layer. In order to keep its use easy, it mainly contains interfaces (or abstract classes) which are implemented by vendor-specific wrappers. The architecture is able to provide the middle layer services also with other technologies such as Restful and Web Services to support the interaction and data exchange among other devices (i.e., mobile devices) and/or industrial systems. In this way, another third application (i.e., mobile application) can access to the central simulation repositories and provide other service over open format data. Actually Floasys Meeting Mobile is under development to provide statistical information about projects during the meetings.

An alternative solution to our architecture could be the introduction of a separate isolation layer for each vendor software. The *support of multiple replaceable* vendor products and the *simulators selection process* requirements impose the introduction of a common isolation layer. In fact, the alternative solution has the following drawbacks: (1) the selection process is performed in the application layer and (2) separate isolation layers means also different APIs, differences that must be handled in the application layer. However, the use of a common isolation layer does not exclude that each wrapper itself is designed with an isolation layer using a proxy pattern.

The extraction of data from closed file format generally is a tricky task and the solution depends on the specific proprietary software and it is strictly coupled with it. The reverse engineering of the binary file content is an extreme solution and we definitively tried to avoid it during Floasys development. Our idea is to interact with the simulator taking advantage of its specific features. Specifically, CFD simulators have an interesting built-in feature: the opportunity to write (or record) a macro to automate tasks within the software. In addition, CFD simulators run "*headless*" without the graphical user interface (GUI) and can execute macros from the command line. It is a built-in feature because every CFD simulation requires and runs on High Performance Computing (HPC) resources. For instance OpenFOAM, an open source CFD software package, is a set of command line tools without GUI so that the aim of many projects [31] both open and commercial is to design a GUI for OpenFOAM. Another CFD simulator is CD-Adapco Star-CCM+, it has a Java-based macro language to automate repetitive tasks. Therefore, Floasys takes advantage of this built-in CFD software feature. In order to extract the data from a closed file format, the specific Floasys Wrapper runs the original simulator and execute a macro within the simulator. The macro reads the simulation content and stores everything in a plain intermediate file that after it is managed by Floasys platform. Floasys reads this plain intermediate file, transforms it to a common open format creating a XML document stored in the central open repository.

In order to meet extensibility and modularity requirements (Req. 7), the server is based on a pure plug-in architecture [32]. A plug-in can provide well-defined hook points called *extension points* to define and describe the way to extend its functionality. Other plug-ins (or modules) can add new functionalities implementing

[2]Edgewall Software Trac official web site: `http://trac.edgewall.org/`

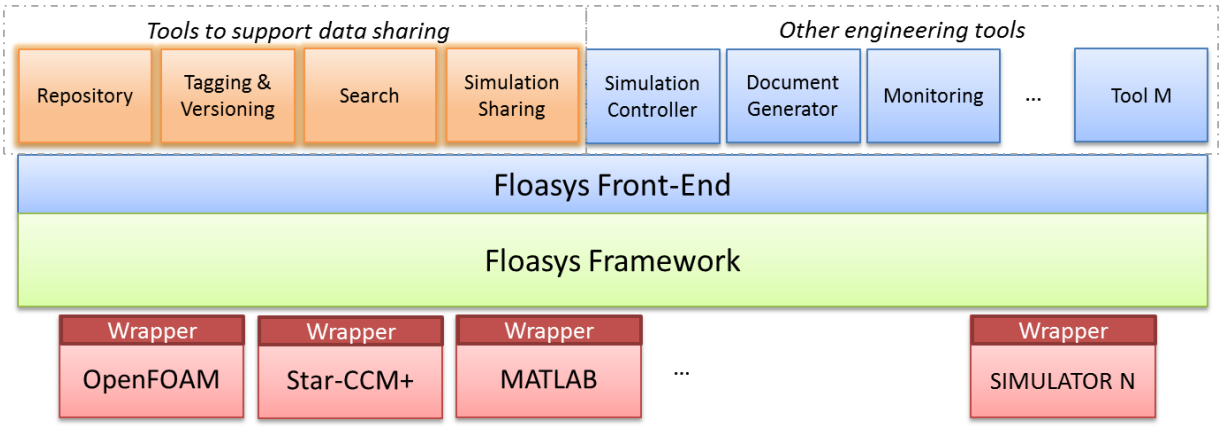

FIG. 4.2. *Floasys Server-side architecture.*

an extension point. In addition, a module can be replaced with another equivalent implementation also at run-time. The Floasys core provides two extensions points to extend its functionalities: (1) one hook point to introduce new tools in the upper layer and (2) another hook point for new wrappers. In this way, the following opportunities exist for the final customers: 1) multiple Floasys instances can be deployed choosing which modules will compose the overall architecture in according to the industrial needs; 2) the industry can identify exactly which modules contain their specific know-how; 3) each company can decide to invest money for the development of its own internal modules to customise Floasys and meet specific internal requirements; 4) in according to Eclipse Public License [20] (EPL), each plug-in can be released open sources or with a closed license.

Floasys has two kind of modules: wrappers on bottom to collect data and tools on top to provide engineering features (Fig. 4.2). An interesting Floasys extension planned for the future is to develop a wrapper that collects experimental data (e.g., wind tunnel experimental data, engine test bed). This is a challenging goal but the advantage would be a central repository that contains both simulation and experimental in open format supporting the comparison among them. An important task is the validation of simulation results and the comparison among the computer results and experimental data is very important.

Floasys relies on mainstream technologies. The server-side components are Java servlet-based. Floasys is developed upon Eclipse Remote Application Platform (RAP) that "*uses standard servlet technology and runs on any JEE servlet container*" [21]. Therefore, the outcome of the deployment phase is a Web application ARchieve (WAR) file that is deployed on a JEE servlet container (e.g., JBoss or Tomcat). This software stack can be installed upon any operating system (e.g., Mac, Windows or Linux). Actually in according to the industrial internal policies, the server is a Red Hat Linux distribution with JBoss[3] but any other Linux distribution can be used.

4.3. Simulation Model: Managing Simulator Differences. Floasys aims to collect data from multiple different simulators that often use closed file formats. A lack of interoperability among CFD software exists (see Section 2) so Floasys must directly handle these heterogeneities. Heterogeneities among vendor products are both syntactic and semantic. The syntactic heterogeneity concerns the vendor product APIs differences or the way to interact with them trough command line. The architecture has an isolation layer (Floasys Framework in Fig. 4.2) to face these syntactic differences that remain within the simulator wrappers and one common API has provided to upper layers. Semantics and data heterogeneities deal with data differences: software are often similar but they use different concepts. This issue becomes evident when architectures try to "*support the concurrent use of multiple infrastructures, transparently*" [8]. Floasys introduces an intermediate common representation for simulation data called **Simulation Data-Model**. It is based on a tree-like data structure as CGNS [33] format. In order to be reusable, it consists mainly of interfaces and abstract classes. In

[3]Red Hat JBoss official web site: `http://www.jboss.org/`

addition, Floasys provides a basic implementation based on the composite design pattern [34]. Each wrapper (architecture bottom layer Fig. 4.2) knows how to interact with a specific simulator and can extract data from a closed file format. The same wrapper is responsible to create the Simulation Data-Model instancing the basic implementation and translating simulation content in nodes of Data-Model. The Simulation Data-Model is serialisable. Floasys serialises the Simulation Data-Models in XML documents that are indexed using Solr and stored in a Subversion repository. Floasys uses Java XStream [35] Library to serialize Simulation Data-Model in XML. This Data-Model is very powerful because Floasys can enrich the original data adding meta-data as a new node of the tree structure. Both Floasys Framework and wrappers can add metadata over data inserting additional nodes in the tree (i.e., documents, automatic extracted information) during extraction phase. Also users can enrich the Data-Model providing tags and comments through repository tool that become nodes in Data-Model. All the information stored in Simulation Data-Model can be used during within the Search Tool to find simulations.

The advantages of our intermediate Simulation Data-Model representation are: (1) metadata over data adding custom nodes, (2) serialisation in open format such as XML, (3) decoupling of wrappers from tools so it is possible to replace a wrapper limiting changes to upper layers and (4) opportunity to compare results that came from simulators with the results that came from the experiments with real prototypes in future. Finally, we experienced a great advantage of using a Data-Model during Floasys development and for the stakeholders after. Using the Data-Model has the advantage to use the Floasys front-end without simulators. The idea is to have a dummy simulator that reads data from the XML file and provides them through the described architecture as a real simulator. This is a cost-saving in terms of HPC resources and available simulator licenses for closed software. Considering the 3D geometry complexity, to open a simulation file, engineers access to a computer cluster using a software license that are fixed by the project budget. Therefore, the requirement to avoid data lock-in leads to a cost-saving feature.

4.4. Simulation Data Centralisation, Version Control and Data Indexing. The architecture integrates multiple simulators, collects and centralises simulation data. Each simulation contains textual, numerical (e.g., results), images and geometrical data. Floasys extracts all simulation data embedded in closed file format and stores them in open format files. The textual and numerical data are stored in XML files in according to the Simulation Data-Model and are committed to the Subversion repository. These XML files are relatively small (MB) so they are easily managed by the Subversion repository. Obviously, most Subversion operations are recursive but Subversion 1.5 introduced the sparse directories [36] (or shallow checkout) to checkout a portion of the working directory with the freedom to get more files and directories later [36]. Therefore, Floasys relies on the shallow checkout to get a partial group of XML files. Floasys can use multiple Subversion servers to accommodate future needs. Version control granularity concerns the specific simulation file. In this way, simulation XML files can be distributed among multiple Subversion servers. Floasys architecture has designed to store the SVN URL within the Solr search engine during the indexing phase. Hence, when the user search a simulation and gets the search results, for each result there is the SVN URL to a specific Subversion repository. Hence, every time Floasys exactly knows the Subversion server used to store the open format XML document. In addition, in order to provide high search performance, the generated simulation XML files are indexed using Apache Solr [25]. Apache Solr provides extensions, configuration, infrastructure and programming languages bindings around Apache Lucene. In according to the official documentation [25], Apache Solr is is highly reliable, scalable and fault tolerant, providing distributed indexing, replication and load-balanced querying, automated failover and recovery, centralized configuration and more. In particular, Apache Solr can be run in a standalone configuration or it is possible to setup a cluster of Solr servers through SolrCloud to combine fault tolerance and high availability as well as scalability using replication and distributed indexing dividing the index into partitions called shards.

Floasys does not use the Subversion repository for the geometrical data because they are very huge (GB). A simulation contains mainly two meshes (geometrical data): (1) a *surface mesh* that is the vehicle shapes used to build the (2) *volume mesh* used at solving time to solve the simulation. Floasys extracts only the surface mesh and makes two outputs: a simplified geometry that serves just as overview of the vehicle product (it is fast to retrieve and render with WebGL, see Section 3.3) and a surface mesh file (e.g., STL file). Floasys does not store geometric volume mesh (the most heavy part of a simulation) reducing the overall required amount

of physical space. In this way it saves space on repositories and it is always possible to build volume mesh from surface mesh.

In order to get the simplified 3D geometry version used only for the visualisation on web, Floasys in batch connects to the Matlab server and reduces the original STL surface mesh creating the lightweight version. This simplified version contains all vehicle parts separately. After many attempts the best trade-off between running time and the 3D geometry quality is to use the Matlab `reducepatch` command. The quality of the obtained mesh is assessed asking to CFD analysts. Floasys interacts with Matlab as a black box, it gives in input the original mesh and gets in output the simplified mesh, so in future we could replace Matlab with another system.

The proposed solution has an interesting advantage. XML files store the most important and useful simulation data including a simplified 3D geometry. Therefore, users can open the XML files using the repository tool and access to all simulation data without the original software and without the HPC resources. It is a useful feature because sometimes CFD analysts need to open simulations to consult data, in this way no proprietary software license nor HPC resources are used.

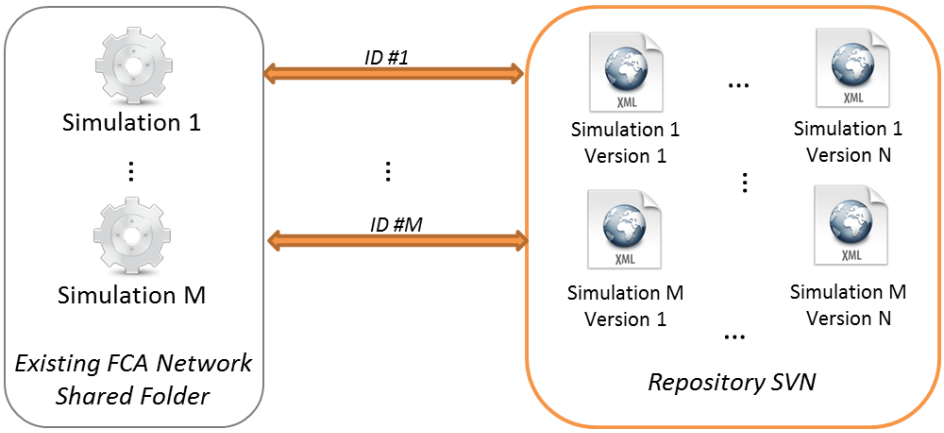

FIG. 4.3. *Simulation data versioning.*

For each simulation file (left-side of Fig. 4.3) stored in closed file format, an XML file exists in the SVN repository (right-side of Fig. 4.3) that contains extracted simulation data and metadata in open format. In addition, each XML file is indexed using Apache Solr [25].

Each XML file is always linked with its original simulation file using an unique ID. In this way, the users can always get the original simulation following the provided link. Floasys generates a unique ID for each simulation file and stores it with metadata in the XML file. The ID is based on the original simulation file content and path. This solution has the following advantages. Floasys does not change the simulation file content to add other information such as the ID. It performs search operations using indexed XML content getting high performances and providing version control for them. Another alternative solution is to add metadata directly to simulation files avoiding the creation of XML files. This solution has been discarded because has the following drawbacks: 1) it is difficult to find available and unused fields in the simulation files; 2) the simulation files are still stored in closed file format, so the solution is vendor software specific; 3) the metadata management requires the access to files through the vendor software using HPC resources due the geometry data and 4) it is difficult to provide version control over simulation files because they takes up to ten gigabytes.

From implementation point of view, two Java libraries have been used (Fig. 4.1): SolrJ to interact with the Solr Server and SVNKit to commit and update data to Subversion repository. Our solution meets also other industrial constraints, such as the impossibility to move existing files and folders or to store them within a database. Finally, the solution must be independent by the specific simulator, so it can not store metadata within the simulation files, also because files are in closed file format.

5. Related Works. Aberdeen Group conducts market research studies to help businesses worldwide to improve performance[4]. They use a research methodology called P.A.C.E. to classify companies in three categories: best-in-class, average and laggard. Then they identify and compare companies using the internal and external pressures, their capabilities and the actions used to face the market challenges. The market research "*Getting Product Design Right the First Time with CFD*" [2] by Aberdeen Group studied the experience of 704 companies that perform simulations to design their products. Specifically, they use the Computational Fluid Dynamics (CFD) simulations to design the products. Their leading market research question is how the CFD simulations impact the product design and which are the key advantages of using them. The white paper includes a list of "actions" that are the steps to perform in order to increase the competitiveness of the companies on the market. Some of the actions are: *capture and document best practices for conducting simulations*, *centrally manage the simulation results and the best practices*, *take advantage of predefined wizards or templates to guide less experienced users*. The market research provides some starting points that must be further investigated, such as "*promote the collaboration*" among engineers, *ensure the right people have access to the results* and *offer version control*. Obviously, the market research does not discuss the technical solutions to achieve these actions.

We had the opportunity to work closely with professionals in Fiat Chrysler Automobiles (FCA) who use CFD simulations to design vehicle products. Our work further investigates the collaborative requirements of dispersed teams and co-located engineers gathered using interviews and a survey. Here, we analyse the survey requirements results enriching them with the stakeholders observations and feedback. Our work contributes also with technical solutions to meet the reported requirements. In [4], authors conducted a survey *to understand the needs and perception of practitioners about the Cloud-based simulation (CBS)*. In their survey results come to light the need to share, store and retrieve models in CBS.

Many Web-based platforms have been created over the years to support Computational Fluid Dynamics. The "*e-Science Aerospace Integrated Research System*" (e-AIRS) [6] is an educational Web portal developed in Korea to help students to understand the aerodynamic simulation process [37]. EDISON_CFD [38] is the e-AIRS improvement in terms of stability, faster data response time and waiting time [39, 40]. Such systems have remarkable differences with our use case requirements and with Floasys. The systems target is the first difference, both e-AIRS and EDISON_CFD have an educational target, instead Floasys aims to industrial sectors (e.g., automotive sector). The e-AIRS target is educational and therefore it has been used in undergraduate and graduate classes. This have an impact on the integrated tools, that is the other difference. e-AIRS integrates custom in-house meshing tools and solvers. It operates with its own Fortran-based in-house CFD solvers [6]. Industries use widely adopted and validated CFD software, so Floasys platform aim is to integrate existing both commercial and open source solvers (Req. 6). In addition, the meshing is very important because it impacts on simulation quality results and running times. e-AIRS adopts a custom software called e-AIRSmesh to mesh the geometry storing the mash in a specific custom file format. Each CFD simulator works with a specific mesh topology. A Floasys requirement is to integrate multiple industrial adopted and validated CFD solvers (Req. 6). Industries have assistance contracts with CFD software vendors, so industrial platforms can not ignore their integration. In addition the aim is to avoid Vendor Lock-In adopting open format data.

Many other platforms proposed to manage simulations on HPC resources but they do not focus on collaboration among engineers. For example, a Web-based system for Management of CFD simulations for Civil Engineering was proposed with the goal to develop tools for civil engineers who are not CFD experts but need to perform CFD analysis [38]. It allows the "*dispatching and controlling of long-running simulations*" [38]. The system targets are civil engineers and CFD beginner users. The system was tested with a group of students in civil engineering class. The main differences concern the system end-user target and the correlated requirements to achieve. Our system target is automotive industry where CFD analysts need to collaborate, share data, result and knowledge, simulation data and result centralisation with the aim to promote collaboration. An interesting emerged common requirement is the need to use templates both for expert and beginner users. The nature of CFD simulations with high number of parameters to consider forces the creation of standard templates both to support beginner and expert users.

Another research avenue comes from the Semantic Web field. Many works in literature proposed software

[4]Aberdeen Group official web site `http://www.aberdeen.com/`

platforms for modelling and simulation. Simantics [41] is ontology based modelling; it uses ontologies to semantically describe the simulation model and the data. The two mainly applications that have built on Simantics platform are: the proprietary Apros6 for power plant M&S and an open source Simantics System Dynamics Tool based on Melodica language and the OpenMelodica environment. The Simantics's [41] developers are working on the integration of OpenFoam, an open source CFD software package.

6. Conclusions and Future Works. In this paper, we were able to identify key collaborative requirements for CFD design through the use of stakeholders interviews and a user survey. In addition we were able to address these requirements with an integrated, extensible and modular architecture. In this way, the paper provides the solutions and the technologies able to address the collaborative requirements. Requirements, solutions and technologies are tracked through the paper and their links are depicted in the Figure 2.6. Floasys is Web-based platform designed and developed to meet the collaborative requirements and is the industrial prototype currently under testing and evaluation in FCA.

Ideas behind Floasys, such as the integrated, extensible and modular architecture, could be adopted also in other contexts. The great opportunity to have different modules to plug in the architecture allows the deployment of a system tailored to engineers needs and development of some custom modules to embed team know-how. The solution to integrate existing engineering software and extract data from closed file format enables the creation of value added services over open format industrial data. In addition, large industries, independently by the sector, have multiple geographically distributed teams so, the collaboration around open format data and the sharing of data at different granularity and aggregation are great features. All features that could boost the industry competitiveness.

Floasys relies on mainstream open source solutions and its architecture is made integrating widely used existing enterprise technologies. The architecture can be divided into four main uncoupled parts: (1) simulators wrappers that communicate with the simulator software to get the simulation data and transform them in XML open format, (2) the version control repository for the XML files (e.g., SVN), (3) an enterprise search engine to index, cache and search the XML documents (e.g., Apache Solr), and (4) the central web server that provides the Web content (e.g., JBoss servlet container). Actually, we choose Apache Solr because it can scale using SolrCloud, Subversion because Floasys supports multiple SVN repositories and a mainstream servlet container. As future works, we have planned a controlled benchmark test to quantitatively assess and evaluate the Floasys performance, reliability and robustness. In addition, we are planning an evaluation study to analyse the usability of the Floasys user interface, and the user satisfaction when interacting with it [42, 43]. The user acceptance of the software will be investigated as well [44].

Finally, an interesting future work is the opportunity to link our SVN that contains simulation data in open format with an internal social network and enable the discussion on artefacts [45]. The aim is to understand and evaluate the benefits of using the social in the field of industrial CFD simulations.

REFERENCES

[1] D. Sörensen, *The automotive development process*. Springer, 2006.
[2] C. K.-R. Michelle Boucher, *Getting Product Design Right the First Time with CFD*, 2011.
[3] D. H. Michelle Boucher, *Engineering Envolved: Getting Mechatronics Performance Right The First Time*, 2008.
[4] S. Onggo, S. Taylor, and A. Tulegenov, *The need for cloud-based simulation from the perspective of simulation practitioners*, Proceedings of the Operational Research Society Simulation Workshop (SW14), 2014.
[5] C. Gargiulo, D. Pirozzi, V. Scarano, and G. Valentino, *A platform to collaborate around CFD simulations*, 23rd IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2014), Parma, Italy, 23-25 June, 2014, pp. 205–210.
[6] J. Moon, C. Kim, Y. Kim, and K. W. Cho, *CFD Cyber Education Service using Cyberinfrastructure for e-Science*, in Fourth International Conference on Networked Computing and Advanced Information Management (NCM'08), 2008, vol. 2. pp. 306–313.
[7] V. Bertram and P. Couser, *Aspects of Selecting the Appropriate CAD and CFD Software*, 9th Conference on Computer and IT Applications in the Maritime Industries, Gubbio, Italy, 2010.
[8] W. H. Brown, R. C. Malveau, and T. J. Mowbray, *AntiPatterns: refactoring software, architectures, and projects in crisis*, 1998.
[9] Cunningham & Cunningham, Inc., *Anti-Pattern*, [Online]. Available: http://c2.com/cgi/wiki?AntiPattern, checked on 19/01/2015.

[10] J. Vlissides, R. Helm, R. Johnson, and E. Gamma, *Design patterns: Elements of reusable object-oriented software*, Reading: Addison-Wesley, vol. 49, p. 120, 1995.
[11] M. Perry and T. Margoni, *Floss for the canadian public sector: open democracy*, IEEE Fourth International Conference on Digital Society (ICDS'10), pp. 294–300.
[12] R. Shah, J. Kesan, and A. Kennis, *Lessons for open standard policies: a case study of the Massachusetts experience*, in Proceedings of the 1st international conference on Theory and practice of electronic governance, 2007.
[13] V. Varma, *Software Architecture: A Case Based Approach*. Pearson Education India, 2009.
[14] S. Sakr, A. Liu, D. M. Batista, and M. Alomari, *A survey of large scale data management approaches in cloud environments*, IEEE Communications Surveys & Tutorials, vol. 13, no. 3, pp. 311–336, 2011.
[15] C.-W. Chang, P. Liu, and J.-J. Wu, *Probability-based cloud storage providers selection algorithms with maximum availability*, IEEE International Conference on Parallel Processing (ICPP), pp. 199–208, 2012.
[16] B. W. Fitzpatrick and J. Lueck, *The case against data lock-in*, Queue, vol. 8, no. 10, 2010.
[17] A. Geraci, F. Katki, L. McMonegal, B. Meyer, J. Lane, P. Wilson, J. Radatz, M. Yee, H. Porteous, and F. Springsteel, *IEEE standard computer dictionary: Compilation of IEEE standard computer glossaries*. IEEE Press, 1991.
[18] B. Bruegge and A. H. Dutoit, *Object-Oriented Software Engineering Using UML, Patterns and Java-(Required)*. Prentice Hall, 2004.
[19] J. Humble and D. Farley, *Continuous delivery: reliable software releases through build, test, and deployment automation*. Pearson Education, 2010.
[20] *Eclipse public license*. [Online]. Available: http://www.eclipse.org/legal/epl-v10.html, checked on 19/01/2015.
[21] *Eclipse RAP Remote Application Platform*, [Online]. Available: http://eclipse.org/rap/, checked on 19/01/2015.
[22] J. Rama and J. Bishop, *A survey and comparison of cscw groupware applications*, in Proceedings of the 2006 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries, 2006.
[23] X. Chen and K. Kasemir, *Bringing control system user interfaces to the web*, TUPPC078, ICALEPCS, vol. 13.
[24] G. Wang, *Improving data transmission in web applications via the translation between XML and JSON*, in Third International Conference on Communications and Mobile Computing (CMC), 2011, pp. 182–185.
[25] A. Solr, *Apache software foundation Solr*, 2014. [Online]. Available: http://lucene.apache.org/solr/, checked on 19/01/2015.
[26] R. Kuć, *Apache Solr 4 Cookbook*. Packt Publishing Ltd, 2013.
[27] D. Smiley and D. E. Pugh, *Apache Solr 3 Enterprise Search Server*. Packt Publishing Ltd, 2011.
[28] T. A. Howes, M. C. Smith, and G. S. Good, *Understanding and deploying LDAP directory services*. Addison-Wesley Longman Publishing Co., Inc., 2003.
[29] B. Arkills, *LDAP directories explained: an introduction and analysis*. Addison-Wesley, 2003.
[30] P. Tahchiev, F. Leme, V. Massol, and G. Gregory, *JUnit in action*. Manning Publications Co., 2010.
[31] *OpenFOAM GUIs* [Online]. Available: http://openfoamwiki.net/index.php/GUI, checked on 19/01/2015.
[32] D. Birsan, *On plug-ins and extensible architectures*, Queue, vol. 3, no. 2, Mar. 2005.
[33] T. H. Christopher L. Rumsey, Bruce Wedan and M. Poinot, *Recent Updates to the CFD General Notation System (CGNS)*, 50th AIAA Aerospace Sciences Meeting, 2012.
[34] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.
[35] J. Walnes, J. Schaible, M. Talevi, G. Silveira, *et al.*, *XStream*, [Online]. Available: http://xstream.codehaus.org, 2011, checked on 19/01/2015.
[36] C. M. Pilato, B. Collins-Sussman, and B. W. Fitzpatrick, *Version control with subversion*. O'Reilly Media Inc., 2008.
[37] J. Moon, K. W. Cho, S.-H. Ko, J.-H. Kim, C. Kim, and Y. Kim, *A Cyber Environment for Engineering Cyber Education*, in IEEE Fourth International Conference on eScience, 2008, pp. 532–539.
[38] S. Lee and C. Kim, *Development and utilization of online computational environment for education and research in fluid engineering*, 2013.
[39] Y. Jung, J. Moon, D. Jin, B. Ahn, J. Seo, H. Ryu, O. Byeon, and J. Lee, *Web simulation service improvement on EDISON_CFD*, Computer Science and Technology, 2012.
[40] Y. J. Jung, J. Moon, D.-S. Jin, B.-Y. Ahn, J. H. Seo, H. Ryu, O.-H. Byeon, and J. R. Lee, *Performance Improvement for Web based Simulation Service on EDISON_CFD*, 2013.
[41] *Simantics platform*, [Online]. Available: https://www.simantics.org/simantics/about-simantics/simantics-platform/, checked on 19/01/2015.
[42] *Questionnaire for user interface satisfaction*. [Online]. Available: http://oldwww.acm.org/perlman/question.cgi?form=QUIS, checked on 19/01/2015
[43] *Computer system usability questionnaire*, [Online]. Available: http://oldwww.acm.org/perlman/question.cgi?form=CSUQ, checked on 19/01/2015.
[44] F. D. Davis, *Perceived usefulness, perceived ease of use, and user acceptance of information technology*, MIS quarterly, pp. 319–340, 1989.
[45] D. Malandrino, I. Manno, A. Negro, A. Petta, V. Scarano, and L. Serra, *Social team awareness*, in 9th International Conference Conference on Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), 2013, pp. 305–314.

Appendix A. Requirements elicitation Survey.

The survey aims to identify the most important requirements for a collaborative simulation platform to support the engineering activities. Survey is confidential and all data will be processed in aggregated way. Thank you for your time and your advice. At the end we will provide you the survey results and considerations.

Your experience.

Q1. Which is your role in the company?
- CFD analyst
- Technical Manager
- Performance Engineer

Q2. Which is your place of work?
- *(FCA) Pomigliano D'Arco (Naples)*
- *(FCA) Orbassano (Turin)*

Q3. How many years you spent working in the CFD field?
- *(write the number of years)*

Q4. How many simulations do you perform per year?
- *(write the number of simulations per year)*

Collaboration among analysts and data sharing.

Q5. In my office I daily work with a number of analysts equal to
- *(write the number of analysts)*

Q6. I daily work with a number of analysts in a different place equal to
- *(write the number of analysts*)

Q7. On average, the geometries file size in average is
- *(write the geometry file size)*

Q8. On average, the simulations file size in average is
- *(write the simulation file size)*

Q9. In order to exchange geometries and simulations files I usually use

E-mail:	(Never)	1	2	3	4	5	6	7 (Always)
Chat:	(Never)	1	2	3	4	5	6	7 (Always)
Phone:	(Never)	1	2	3	4	5	6	7 (Always)
FTP:	(Never)	1	2	3	4	5	6	7 (Always)
Ask to a colleague:	(Never)	1	2	3	4	5	6	7 (Always)
Internal Platform:	(Never)	1	2	3	4	5	6	7 (Always)

Q10. In order to exchange documents I usually use

E-mail:	(Never)	1	2	3	4	5	6	7 (Always)
Chat:	(Never)	1	2	3	4	5	6	7 (Always)
Phone:	(Never)	1	2	3	4	5	6	7 (Always)
FTP:	(Never)	1	2	3	4	5	6	7 (Always)
Ask to a colleague:	(Never)	1	2	3	4	5	6	7 (Always)
Internal Platform:	(Never)	1	2	3	4	5	6	7 (Always)

Data centralisation and simulation data search.

Q11. I follow some rules to store simulations and assign the name to their corresponding files

(Never) 1 2 3 4 5 6 7 (Always)

Q12. The information that I store in the simulation file name are *(Multiple choice)*
- *Project Name*

- *Release*
- *Ground Clearance*
- *Revision*
- *Engine*
- *Vehicle Trimming*
- *Date*

Q13. The rules are:
- *Personal Choice*
- *Team Conventions*
- *Fixed imposed rules*

Q14. I follow the rules over time
(Never) 1 2 3 4 5 6 7 (Always)

Q15. The opportunity to link other information (e.g., tags) to files could be:
(Useless) 1 2 3 4 5 6 7 (Useful)

Q16. In order to find simulation files I usually use

My mind:	(Never) 1 2 3 4 5 6 7 (Always)
Free directory navigation:	(Never) 1 2 3 4 5 6 7 (Always)
Windows Find Tool:	(Never) 1 2 3 4 5 6 7 (Always)
See the file name:	(Never) 1 2 3 4 5 6 7 (Always)
Open the simulation:	(Never) 1 2 3 4 5 6 7 (Always)
File History:	(Never) 1 2 3 4 5 6 7 (Always)
Unix Find Tool:	(Never) 1 2 3 4 5 6 7 (Always)
Ask to a coworker:	(Never) 1 2 3 4 5 6 7 (Always)

Q17. I have a tool to search simulations according to their content
(Never) 1 2 3 4 5 6 7 (Always)

Q18. A tool to support the search operations based on simulation data could be:
(Useless) 1 2 3 4 5 6 7 (Useful)

Simulation data versioning.

Q19. I have a tool to show the simulation's modification over time
(Never) 1 2 3 4 5 6 7 (Always)

Q20. A tool to show the simulation revisions could be
(Never) 1 2 3 4 5 6 7 (Always)

Used simulators.

Q21. During my work I use these simulator software (multiple choices):
- *Star-CCM+*
- *OpenFOAM*
- *SolidWorks*
- *PowerFlow*
- *CFD++*

Edited by: Giacomo Cabri
Received: September 15, 2014
Accepted: January 5, 2015

DOI 10.12694/scpe.v15i4.1054
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 15, Number 4, pp. 331–343. http://www.scpe.org

INVESTIGATION ON THE OPTIMAL PROPERTIES OF SEMI ACTIVE CONTROL DEVICES WITH CONTINUOUS CONTROL FOR EQUIPMENT ISOLATION

MICHELA BASILI* AND MAURIZIO DE ANGELIS†

Abstract. The paper treats the semi active isolation of a single equipment, acceleration sensitive, by means of a variable elastic control device. A numerical study on a single degree of freedom (SDOF) structural model equipped with a continuously variable elastic device subjected to harmonic input is presented. The utilized control algorithm is derived by the Lyapunov method and specialized in order to obtain instantaneous optimality. In order to minimize the dynamic response of interest, i.e. the equipment absolute acceleration, some parameters that define the algorithm and the device are conveniently selected. The purpose of the paper is to investigate the optimal isolation properties of semi active variable stiffness devices with continuous control across the whole frequency spectrum. The performances of the isolated equipment are evaluated in terms of absolute acceleration transmissibility. Semi active continuous control is compared with semi active ON-OFF mode and conventional linear passive control. Results show that it is possible to choose conveniently the parameters regulating semi active continuous control in order to limit the absolute acceleration transmissibility at all the frequencies. In literature from problems concerning vibration isolation, transmissibility is alternatively defined in terms of absolute acceleration or displacement. Here, absolute displacement transmissibility is also estimated. It is observed that in case of semi active control, there are differences between the two transmissibility representations, and they do not lead to analogous results for evaluating the performance of the control system.

Key words: Equipment isolation, semi active control, continuous control law, absolute acceleration and displacement transmissibility

AMS subject classifications. 34H35, 98C83

1. Introduction. Among the different control strategies, semi active devices appear interesting for vibration mitigation, since they can adjust adaptively their mechanical characteristics (stiffness and/or dissipation parameters) in real time depending on the input and/or the structural response according to a given control law, without adding external energy to the structures, thus not compromising system stability. Numerical and experimental works that focus the attention on the performances of semi active control for the dynamic response of systems with different structural configurations are present in literature, e.g. [1], [2]. The force supplied by semi active devices can vary according to a given control law which can be ON-OFF or continuous. In ON-OFF operation the control devices may assume only one of the two operation states: ON state (device activated) and OFF state (device deactivated). In continuous operation, the mechanical parameters of the semi active devices may continuously assume any value between the given limits. Most of the studies and applications found in the literature concern applications with ON-OFF control, e.g. [1]- [5], whereas only few papers concern continuous semi active control, [6]- [9], for this reason, closer attention will be given to this area in the following investigation.

Equipment of various nature, intended as objects of artistic great value (statues or sculptures), or precision technical equipment, or sensitive and refined medical components in the hospitals, placed in buildings usually need to be protected against vibrations. Among this class of objects, the study is concerned with acceleration sensitive components prone to damage from inertial loading. Passive systems have been proved to be effective and practical, but at the same time might suffer from low-frequency resonances and excessive isolator displacements, if subjected to long period ground motions (e.g. near-fault earthquakes). For this reason, efforts on the utilization of semi active devices for vibration isolation are made; the response transmitted to the equipment can be effectively reduced, and performances appear superior when compared with conventional linear passive control strategy [10]- [12].

This study investigates the base isolation of acceleration sensitive equipment by means of a variable elastic device with continuous control law. A single degree of freedom model is adopted with a harmonic base motion as input motion condition and the attention is focused on the absolute approach (absolute motion with respect to a fixed reference). The continuous control law is derived from the Lyapunov method and specialized in order to obtain instantaneous optimality. The aim of the paper is to investigate the optimal properties of semi active

*Department of Structural and Geotechnical Engineering, Sapienza University of Rome, Italy (michela.basili@uniroma1.it)
†Department of Structural& Geotechnical Engineering, Sapienza University of Rome, Italy (maurizio.deangelis@uniroma1.it)

continuous control varying the frequency ratio between the input and the equipment, in order to cover all the regions of the frequency spectrum. Two parameters, one related to the algorithm and the other related to the device, defined as the stiffness ratio between the device and the equipment, are optimized in order to minimize the equipment acceleration. The performance of the optimized control system is evaluated in terms of absolute acceleration transmissibility curves in order to investigate the behavior across the whole frequency spectrum. Comparisons with semi active ON-OFF and conventional linear passive control are also carried out.

Since in literature from problems concerning vibration isolation, transmissibility is alternatively defined in terms of absolute acceleration or displacement (often interpreted similar), in this paper, absolute displacement transmissibility is estimated as well. The aim of the paper is also to investigate if there are differences between the two transmissibility representations in case of semi active control, and to see if they lead to analogous results for evaluating the performance of the control system.

The paper is organized as follows: section two reports the description of the single degree of freedom model together with the governing equations, section three presents the semi active control law, section four discusses the case study and, finally, section five carries out the results in terms of control system optimization and the performances of semi active continuous control compared with semi active ON-OFF and conventional linear passive control.

2. Description of the model and governing equations. The reference model is represented by a single degree of freedom (SDOF) undamped structural model, base-excited, Fig. 2.1.

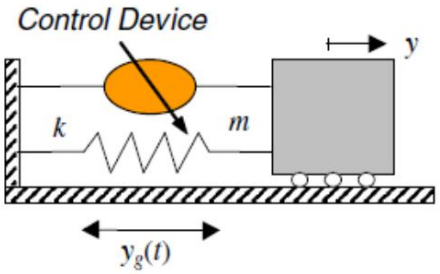

FIG. 2.1. *SDOF reference model.*

The state-space equations of motion, in the relative or absolute state space, can be written as:

$$\begin{aligned} \dot{\mathbf{z}}_{r,a}(t) &= \mathbf{A}\mathbf{z}_{r,a}(t) + \mathbf{B}u(t) + \mathbf{H}_{r,a}w_{r,a}(t) \\ \mathbf{z}_{r,a}(0) &= \mathbf{z}_{(r,a)0} \end{aligned} \tag{2.1}$$

where

$$\begin{aligned} \mathbf{z}_{r,a} = \begin{pmatrix} y \\ \dot{y}_{r,a} \end{pmatrix}; \mathbf{A} = \begin{pmatrix} 0 & 1 \\ -\omega^2 & 0 \end{pmatrix}; \mathbf{B} = \begin{pmatrix} 0 \\ -1/m \end{pmatrix} \\ \mathbf{H}_r = \begin{pmatrix} 0 \\ 1 \end{pmatrix}; \mathbf{H}_a = \begin{pmatrix} -1 \\ 0 \end{pmatrix}; w_r(t) = \ddot{y}_g(t); w_a(t) = \dot{y}_g(t) \end{aligned} \tag{2.2}$$

where ω is the circular frequency of the equipment, y is the relative displacement of the structural mass m with respect to support, y_g is the displacement of the support with respect to a fixed reference and u is the force in the control device. Finally, k is the elastic structural stiffness ($k{=}m\omega^2$) and T_0 is the natural period ($T_0{=}2\pi/\omega$). Absolute response quantities are evaluated by adding the support motion quantities to the corresponding relative ones. The present study refers to a variable stiffness device with control force:

$$u(t) = \lambda(t)m\omega^2[(y(t) - y(t_i)] \tag{2.3}$$

where the relative stiffness $\lambda(t)$ represents the ratio between the device (k_d) and the structural (k) stiffness, whereas $y(t_i)$ is the structural displacement corresponding to the last activation instant t_i of the device. The $\lambda(t)$ parameter has the following physical limitation:

$$0 \leq \lambda(t) \leq \lambda_{max} \tag{2.4}$$

3. Semi active control law. The direct Lyapounov method is used to define the control algorithm. It is based on the definition of an appropriate Lyapounov function and a control law which guarantees the hypothesis of the Lyapounov theorem. The state function, assumed as Lyapounov function, in the relative or absolute approach is defined as:

$$\Lambda = \frac{1}{2}\mathbf{z}_{r,a}^T(t)\mathbf{Q}\mathbf{z}_{r,a}(t) + \int_0^\tau \mathbf{u}^T\mathbf{R}\mathbf{u}(\tau)d\tau \tag{3.1}$$

where $\mathbf{Q}$ is the weighting matrix (symmetrical and, at least, positive semi-definite) of the state and $\mathbf{R}$ (symmetrical and positive definite) weights the control force. The procedure in order to select an optimal control law based on the Lyapounov function defined in Eq. 3.1 is deeply explained in [6] in the case of the relative approach. In this work only the final expression of the optimal control force is reported, valid for the relative and the absolute approach:

$$\mathbf{u}_{opt}(t) = -\frac{1}{2}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{Q}\mathbf{z}_{r,a}(t) \tag{3.2}$$

It is important to remark that the resulting law Eq. 3.2 guarantees only locally (for a given time) the optimality of the control law (i.e. a local minimum of the state function). It is possible to give a physical interpretation of the control process since the Lyapunov function Eq. 3.1 may be also viewed as input energy. In the case of a SDOF system, matrices appearing in Eq. 3.2 may be written as follows:

$$Q = \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{12} & Q_{22} \end{pmatrix}; \mathbf{R} = R \tag{3.3}$$

The non-dimensional parameters and the optimized parameters for the algorithm introduced in [6] are used and, after passages, the optimal control force is:

$$u_{opt}(t) = \frac{m\omega}{\rho}\dot{y}_{r,a}(t) \tag{3.4}$$

For a variable stiffness semi active device with constitutive equation expressed by Eq. 2.3, the variation law of the device stiffness is obtained as follows:

$$\lambda(t) = F[0, \lambda^*(t), \lambda_{max}] \tag{3.5}$$

where for the relative approach the $\lambda^*(t)$ parameter is evaluated as:

$$\lambda^*(t) = \frac{1}{\rho\omega}\frac{\dot{y}_r(t)}{[(y(t) - y(t_i)]} \tag{3.6}$$

and for the absolute approach is:

$$\lambda^*(t) = \frac{1}{\rho\omega}\frac{\dot{y}_a(t)}{[(y(t) - y(t_i)]} \tag{3.7}$$

Let's observe that for $\rho \longrightarrow 0$, $u_{opt} \longrightarrow \infty$ and, by evaluating the sign of this limit, the ON-OFF control law is obtained, [1]. In both approaches, the parameter ρ must be selected.

4. Description of the case study. The study of the forced vibrations of the SDOF system represented in Fig. 2.1 is carried out, having assumed a harmonic base motion. In the Relative Approach (RA), the input is defined in terms of a base acceleration:

$$\ddot{y}_g(t) = A_g \sin(\beta\omega t + \psi) \tag{4.1}$$

where $\beta = \omega_f/\omega$ is the ratio between the input ω_f and the equipment ω circular frequency, A_g is the amplitude and ψ is the phase angle. In this work it is assumed A_g =1, since the structural system is homogeneous of order one [12] and the dynamic response is not dependent of the motion amplitude, and ψ =0.

In the Absolute Approach (AA) the input is defined in terms of a base velocity:

$$\dot{y}_g(t) = V_g \cos(\beta\omega t) \tag{4.2}$$

where $V_g = -A_g/\beta\omega$.

Concerning the control system, an optimal selection of the algorithm parameter ρ and the relative stiffness device parameter λ, will be made by choosing such values that minimize the maximum absolute acceleration. Meantime, it will be checked that the relative structural displacement is be limited.

Two response quantities will be evaluated:

$$\begin{aligned} A &= \frac{\max(a_{a,SAC}(t))}{\max(a_{a,PC}(t))} \\ Y &= \frac{\max(y_{SAC}(t))}{\max(y_{PC}(t))} \end{aligned} \tag{4.3}$$

where A and Y are defined respectively as the maximum values of the absolute acceleration/relative displacement in case of semi active control (SAC) to the maximum corresponding values obtained with classical linear passive control (PC) having assumed a conventional damping ratio of 10%. The response quantities are therefore normalized with respect to the corresponding values assumed in case of conventional linear passive isolation system.

Once having conveniently chosen the algorithm and device parameter ρ and λ respectively, some response functions, all estimated in the stationary response, will be evaluated in order to observe the dynamics of the controlled system and its effectiveness.

The transmissibility factor will be the most important quantity observed. Here, two transmissibility definitions will be utilized, one estimated with the absolute acceleration, TR_a, and the other estimated with the absolute displacement, TR_d, respectively defined as:

$$\begin{aligned} TR_a &= \frac{|a_a(t)|_{max}}{A_g} \\ TR_d &= \frac{|y_a(t)|_{max}}{Y_g} \end{aligned} \tag{4.4}$$

Let's remember that A_g is the acceleration amplitude of the base notion, defined in Eq. 4.1, whereas Y_g represents the displacement amplitude of the base motion.

It is known in literature from problems concerning vibration isolation that transmissibility can be alternatively defined in terms of absolute acceleration or displacement. However, it is important to remark that, only in case of a linear system these two measures are identical. Since the equipment is acceleration sensitive, absolute acceleration transmissibility should be always considered to check the effectiveness of the control strategy. One of the aims of this paper is to investigate what are the main differences between these two representations often interpreted similar in literature. Moreover, the intent is to see if the optimization of the parameters which define the control, leads to the same conclusions in terms of absolute acceleration or displacement transmissibility.

Equipment relative displacement is checked by observing two other quantities. The maximum displacement vibration amplitude, Y_n, that is the ratio of the amplitude of the relative displacement to the static displacement,

and the ratio between the maximum relative equipment displacement Y and the maximum amplitude of the input Y_g, named Y/Y_g, defined respectively as:

$$\begin{aligned} Y_n &= |y(t)|_{max} \cdot \omega^2 \\ Y/Y_g &= \frac{|y(t)|_{max}}{Y_g} \end{aligned} \tag{4.5}$$

4.1. Synthesis of the results obtained for the Relative Approach (RA) versus Absolute Approach (AA). In Ref. [9], the authors studied acceleration sensitive equipment isolated with a semi active variable stiffness device with continuous control. Both relative and absolute approaches have been considered. The algorithm and device parameters were conveniently optimized in order to obtain the smallest equipment acceleration in the range of frequencies typical of the isolation.

In has been shown through absolute acceleration transmissibility curves TR_a versus β, that, in the region where typically isolation strategy is considered ($\beta \geq \sqrt{2}$), semi active continuous control gives superior performances in comparison to the case of semi active ON-OFF control and conventional linear passive control (10% of damping ratio). Moreover, comparing relative and absolute approaches, the paper concluded that, absolute approach should be preferred to relative one since the transmissibility seemed to be better controlled, Fig. 4.1. In particular, if the attention was paid in proximity to the resonance condition and in the region around $\beta \geq \sqrt{2}$, the absolute acceleration transmissibility was well limited if compared with relative approach. The two approaches became equal by increasing β. When β=3, reductions of the absolute acceleration transmissibility up to 80% were observed.

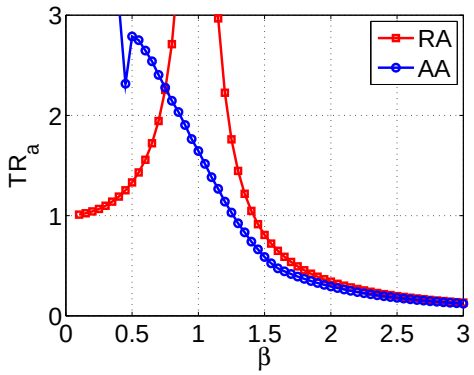

FIG. 4.1. *TR_a varying β for λ=5: continuous SAC, relative approach (ρ=10) and absolute apporach (ρ=0.4).*

5. Results. The aim of this paper is to extend the investigation on the optimal properties of semi active continuous control across the whole frequency spectrum, in order to explore the control system and its effectiveness. The study will focus exclusively on the absolute approach.

The section is organized as follows: first, the optimal parameters ρ and λ at different input frequency ratios β will be selected in order to minimize the absolute acceleration. Then, the absolute acceleration transmissibility curves versus β will be carried out and comparisons among semi active continuous, ON-OFF and passive control will be done. Discussion about utilizing absolute acceleration or displacement transmissibility for acceleration sensitive systems will be debated. Finally, other response functions, defined in Eq. 4.5, to control relative displacement will also be revised. Three regions are meaningful in the frequency spectrum: the region where the input frequency, ω_f, is greater than the system's frequency ω, $\beta > 1$, the region in proximity to resonance condition, β=1, and the region where the input frequency is lower than the system's frequency, $\beta < 1$.

Figure 5.1 depicts the investigation on the properties of the semi active continuous control varying the algorithm and device parameters, ρ and λ, at different values of the frequency ratios β in terms of the dimensionless absolute acceleration and relative displacement response quantities, A and Y respectively. The range

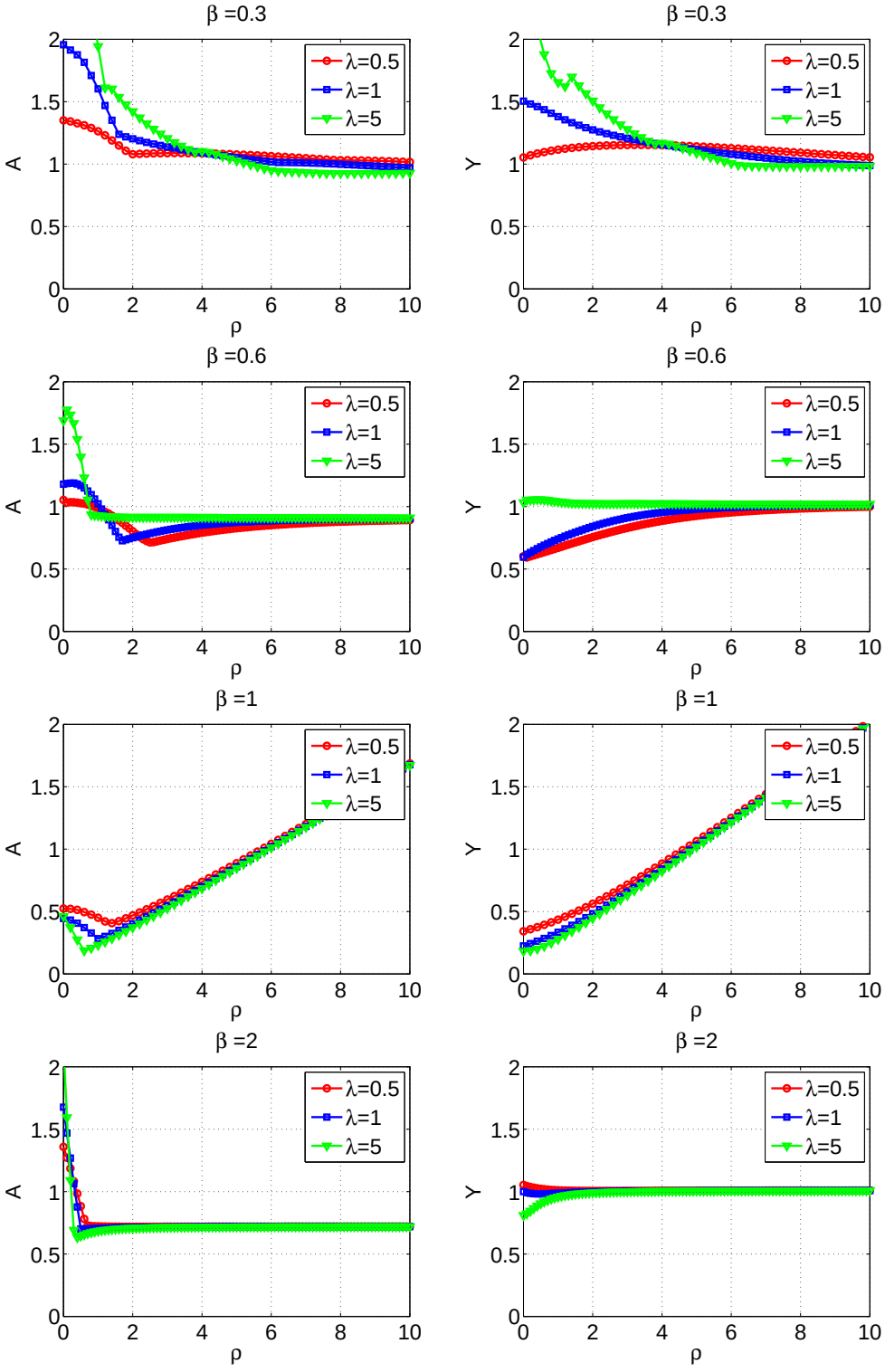

FIG. 5.1. *A and Y varying ρ for different β and λ=0.5, 1, 5.*

of interest of the parameter ρ is among 0 and 10, whereas the λ parameter assumed three values: λ=0.5, 1, 5. Let's observe that, for shortness, λ here means λ_{max}, Eq. 3.5. Optimal ρ and λ are selected as the ones that minimize the absolute acceleration response A.

Since response quantities are dimensionless, when the ordinate values are less than the unity means that response quantities are less respect to conventional linear passive control case. Besides, semi active ON-OFF control corresponds to ρ=0, while semi active continuous control corresponds to $\rho \neq 0$.

Considering the device parameter λ, Fig. 5.1, it can be noticed that, irrespectively of β, the optimal choice is always λ=5. Only in the neighborhood of β about 0.6, the maximum absolute acceleration reduction is obtained for λ=0.5. In fact, for these values of β, the A curves have a well defined minimum at low λ, whereas at λ=5 the curve does not show a minimum and reaches higher values of A.

Considering the algorithm parameter ρ, two different behaviors are observed varying β:

- For $\beta \leq 0.6$, a good choice of ρ can be assuming it at high values, e.g. ρ=10, the effectiveness of semi active continuous control is slightly better than passive control and is always superior to semi active ON-OFF control;
- For $\beta > 0.6$, a good choice of ρ can be assuming it at low values, e.g. ρ=0.7, the effectiveness of semi active continuous control becomes more evident and is always superior to ON-OFF and conventional linear passive control.

By observing the A curves, in proximity to the resonance condition, they have a well defined minimum varying ρ, instead in the other regions the functions decrease until a certain value and then remain constant. Effectively, in [9] an optimal value ρ=0.4 was selected having optimized it only in the region of frequencies typical of the isolation. It can be noticed that the choice ρ=0.7 at $\beta > 0.6$ fits well also in the regions where $\beta \geq \sqrt{2}$, since for such frequencies the acceleration curve is constant increasing ρ (the choice ρ=0.4 or ρ=0.7 is identical in terms of acceleration reduction, see Fig. 5.1 case β=2). However, the frequency ratio zone around β=0.6 where a discontinuity has been observed in the choice of the optimal algorithm and device parameters, deserves a deeper investigation in a future work by the authors in order to explore the peculiar dynamic behavior of the control system.

The relative displacement Y is checked, once selected optimal values of ρ and λ. In general Y is always limited; however, it cannot be identified a systematic trend comparing semi active continuous and ON-OFF control and conventional linear passive control. Relative displacement is not always reduced to the maximum with semi active continuous control.

Figure 5.2 shows absolute acceleration and displacement transmissibility curves, the maximum displacement amplitude Y_n and the displacement ratio Y/Y_g curves versus the frequency ratio β. For comparison purposes, the corresponding curves in the case of semi active ON-OFF control and conventional linear passive control are reported as well. In the case of semi active continuous control the curves are estimated having assumed λ=5, ρ=10 for $\beta \leq 0.6$ and ρ=0.7 for $\beta > 0.6$, whereas, in case of semi active ON-OFF control the curves are estimated for ρ=0 and λ=5.

Since the equipment is acceleration sensitive, absolute acceleration transmissibility TR_a must be primarily observed. Semi active continuous control always leads to better response reduction compared with ON-OFF and passive control. Continuous control can isolate for frequency ratios up to 1.2, but when $\beta \leq 1$ the absolute acceleration transmissibility continues to maintain low values (maximum TR_a is 1.5), which means that the equipment acceleration is sufficiently limited with respect to the base motion. However, it is stressed that acceleration vibration cannot be isolated across the whole frequency spectrum, i.e. absolute acceleration transmissibility cannot be less than the unity across the whole frequency spectrum with any of the assumed three strategies.

Absolute displacement transmissibility curve, TR_d, is different with respect to the absolute acceleration transmissibility curve in the semi active cases; only in case of conventional linear passive control, absolute acceleration and displacement transmissibility are equal.

Therefore, considering absolute acceleration or displacement transmissibility, may lead to different results for evaluating the performance of the control system in the case of semi active control. Displacement vibration can be isolated across the whole frequency spectrum, i.e. absolute displacement transmissibility can be less than the unity across the whole frequency spectrum (semi active ON-OFF control case). By checking the maximum

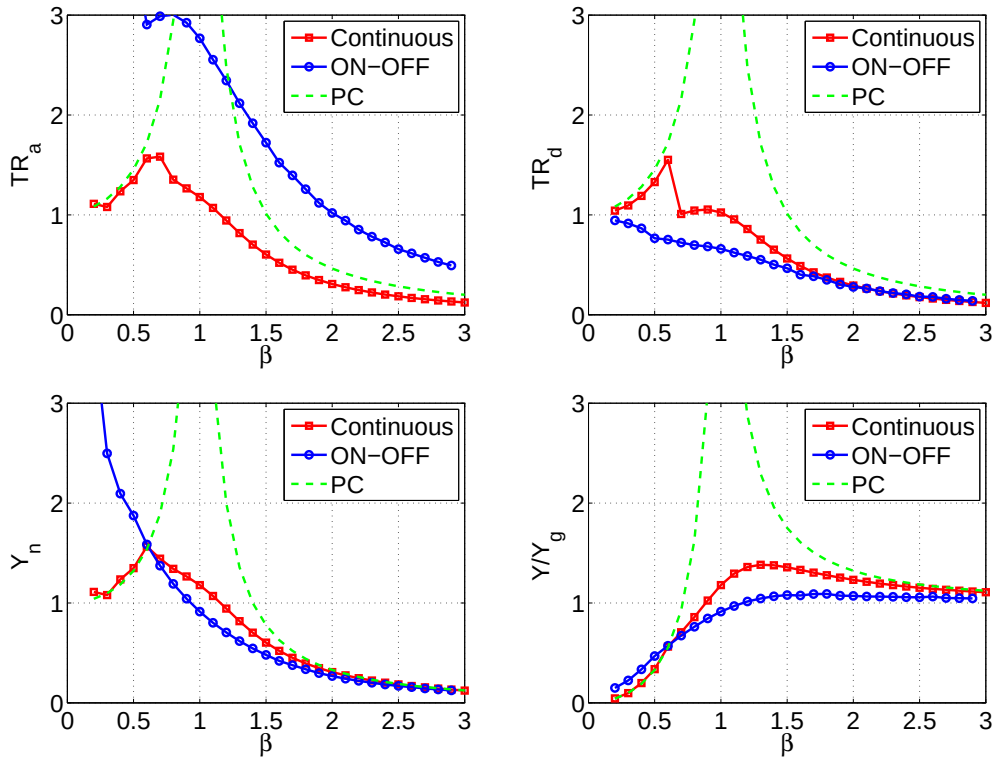

FIG. 5.2. *TR_a, TR_d, Y_n and Y/Y_g varying β for λ=5, semi active continuous (optimized ρ), semi active ON-OFF control (ρ=0), conventional linear passive control, PC, with 10% of damping ratio.*

displacement amplitude Y_n, as $\beta > 1$ it is below the unity only in case of ON-OFF control, when $\beta < 1$ it is greater than one, but still limited only in case of semi active continuous and passive control. By checking the displacement ratio Y/Y_g, when $\beta < 1$, as general trend, it decreases decreasing β and it is almost lower than the unity in the two semi active control strategies, whereas as $\beta > 1$, it tends to the unity as β increases, the convergence is faster in case of semi active ON-OFF control. Curves obtained with semi active continuous and ON-OFF control show light differences, whereas significantly differences are evident with respect to linear passive control, especially in correspondence to the resonance condition.

So far, transmissibility and displacement curves have been obtained having optimized ρ with the frequency ratio, however this action is possible as soon as the input frequency is considered known. If the action and its frequency involved is unknown, such as a natural earthquake, a fixed value for the ρ parameter must be assumed. Figure 5.3 depicts absolute acceleration and displacement transmissibility curves, the maximum displacement amplitude Y_n and the displacement ratio Y/Y_g curves, versus the frequency ratio β for a given value of the ρ parameter. The cases ρ=0.7 (optimal for $\beta > 0.6$) and ρ=10 (optimal for $\beta \leq 0.6$) are reported. For each curve, the results shown in Fig. 5.2, referred to the optimal choice of the algorithm parameter, here can be obtained by crossing the curves at ρ=0.7 and ρ=10. If the expected input action has frequency content mainly in the region over β=0.6 the solution with ρ=0.7 should always be preferred. In fact, acceleration is well limited and displacement is controlled as well.

If the equipment is acceleration sensitive, absolute acceleration transmissibility must be considered and the continuous control law should be preferred to the ON-OFF law or linear passive control. If the equipment is displacement sensitive, absolute displacement transmissibility must be considered and the ON-OFF control law should be preferred to the others.

In order to observe the dynamics of the controlled system with the three strategies, Fig. 5.4 depicts time histories of equipment absolute acceleration for a frequency ratio lower and higher than the resonance frequency

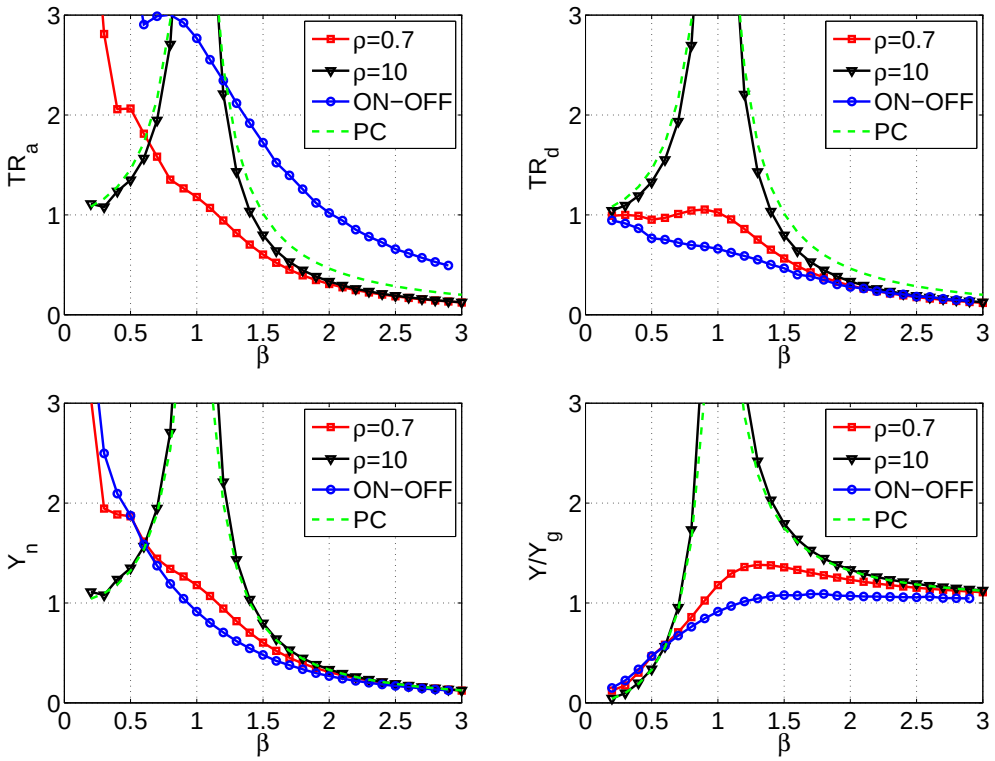

FIG. 5.3. *TR_a, TR_d, Y_n and Y/Y_g varying β for λ=5, semi active continuous (ρ=0.7, ρ=10), semi active ON-OFF control (ρ=0), conventional linear passive control.*

ratio (β=0.8 and β=1.2 respectively), whereas Fig. 5.5 shows, at the same frequency ratios, the variation of the device parameter λ versus time for continuous and ON-OFF control.

The different dynamic behavior in terms of amplitude and shape clearly emerges comparing semi active and passive control; in fact, the acceleration time histories differ from the typical sinusoid of passive control. Semi active continuous control mostly reduces the peaks with respect to the other cases. Semi active ON-OFF case shows the sudden discontinuities in correspondence with the deactivation process, observed in Fig. 5.5 where the time history of the device parameter λ is depicted; instead, continuous case shoots down such peaks in the deactivation process, as a result of the transition in that instants between the maximum and minimum λ being continuous, Fig. 5.5.

By observing the variation of the device parameter λ versus time, Fig. 5.5, in the case of ON-OFF control more than one switch is noticed in one activation and deactivation phase of continuous control for β=0.8, whereas for β=1.2, the frequency of the activation and deactivation phases is almost the same in the two strategies. As known, the transition between the activated (ON) and deactivated (OFF) state is continuous for continuous control and instantaneous for ON-OFF control.

Finally Fig. 5.6 depicts typical absolute acceleration-relative displacement cycles in case of continuous and ON-OFF mode for β=0.8 and β=1.2. A different dynamic behavior in the two cases is observed, and the better effectiveness on the reduction of the absolute acceleration emerges in case of continuous control.

6. Conclusions. This paper treated the topic of base isolation of equipment against vibrations. A SDOF structural model, acceleration sensitive, equipped with a continuously variable elastic device subjected to harmonic input has been discussed, focusing the attention on the absolute approach. The continuous law for the variation of the device parameter has been derived by the Lyapunov method and specialized in order to obtain instantaneous optimality. The aim was to investigate the optimal properties of the semi active continuous control in all the regions of frequency spectrum. Two parameters were optimized in order to minimize the

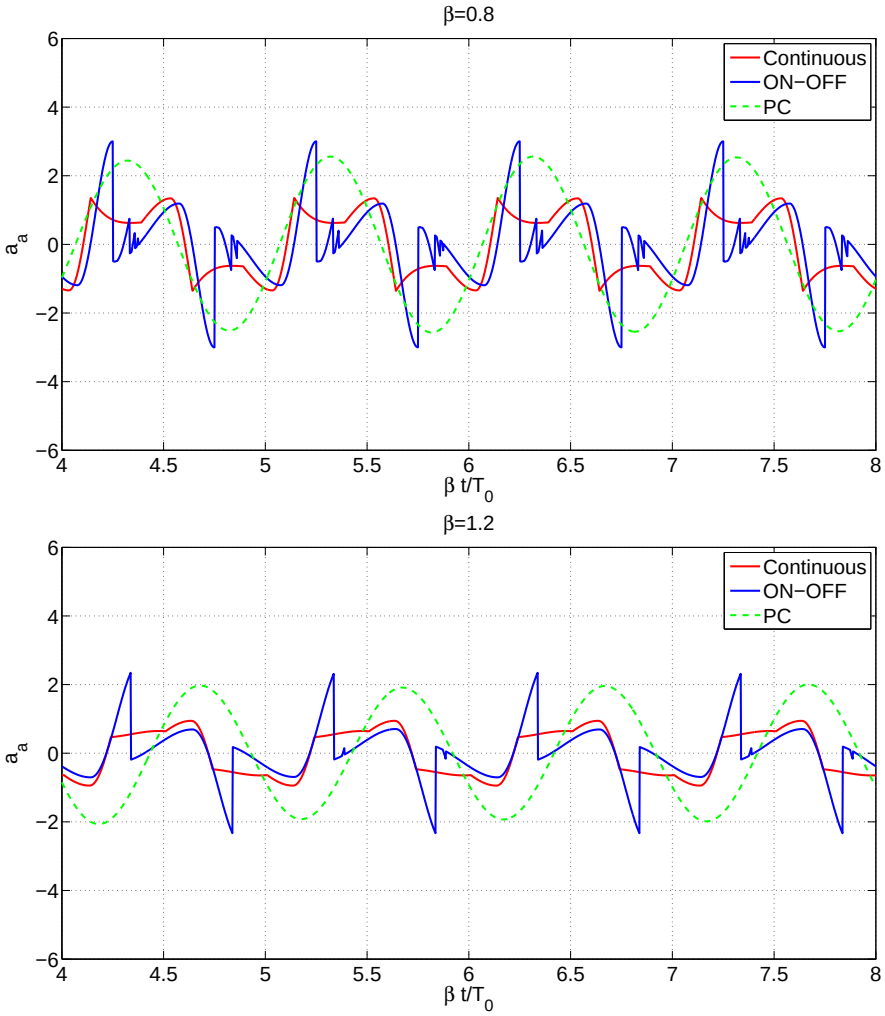

FIG. 5.4. *Equipment absolute acceleration and relative displacement versus time for semi active continuous (ρ=0.7, λ=5), ON-OFF (ρ=0, λ=5), and passive control for β=0.8, 1.2.*

equipment acceleration: ρ related to the algorithm and λ, the stiffness ratio between the device and the equipment, related to the device. It emerged that for the algorithm parameter ρ, two different behaviors are observed varying β: for $\beta \leq 0.6$, a good choice of ρ can be assuming it at high values, whereas for $\beta > 0.6$, a good choice of ρ can be assuming it at low values. For the device parameter λ it was noticed that, irrespectively of β, the optimal choice was almost always to set it at its maximum value in the ON state. This result implies that as the difference among the operational states of the semi active device is greater as the effectiveness of the control system increases. Once optimized the parameters which govern the control system, the absolute acceleration and displacement transmissibility curves, TR_a and TR_d, were evaluated versus β together with the maximum displacement amplitude Y_n and the displacement ratio Y/Y_g. The performance of the continuous semi active control has been evaluated in comparison to semi active ON-OFF and conventional linear passive control.

The performance of the optimized control system was evaluated in terms of absolute acceleration transmissibility curves in order to investigate the behavior across the whole frequency spectrum. It was shown that semi active continuous control always led to better response reduction compared with ON-OFF and linear passive control. It isolated for frequency ratios up to 1.2, still maintaining, for lower frequency ratios, equipment acceleration sufficiently limited with respect to the base motion.

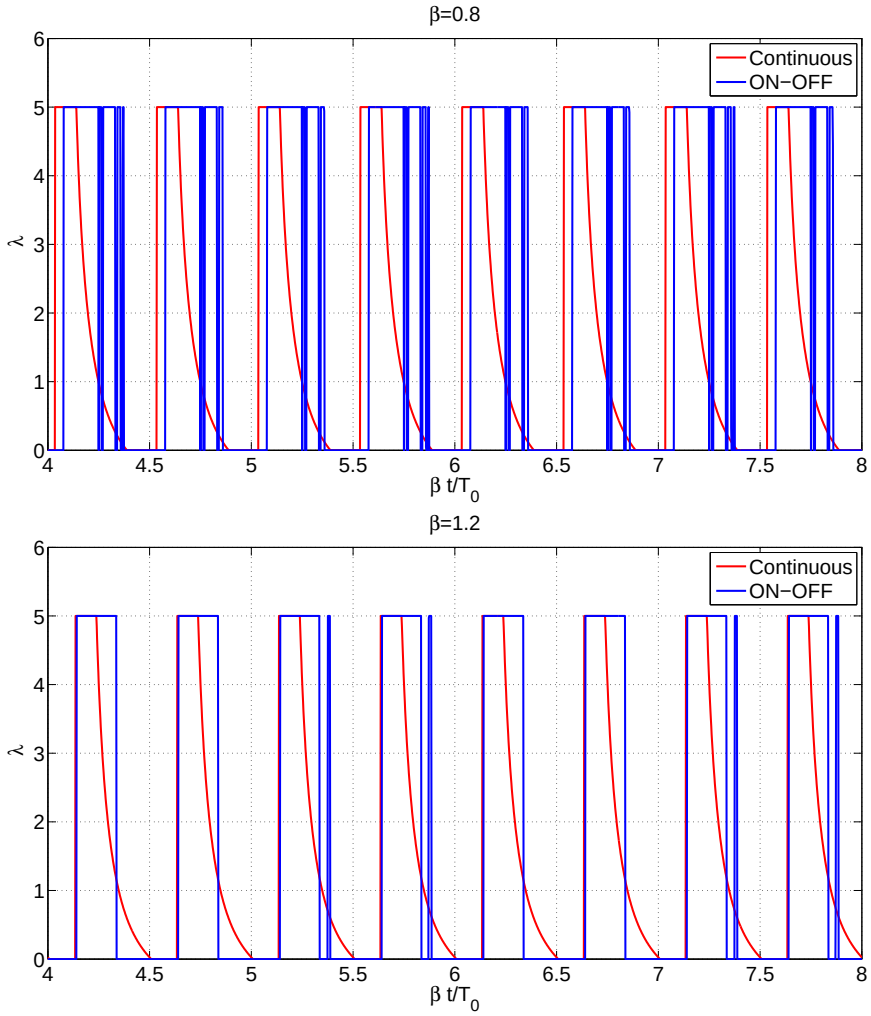

FIG. 5.5. *Variation of the device parameter λ vesrus time for semi active continuous (ρ=0.7) and ON-OFF (ρ=0) control for β=0.8, 1.2.*

In literature from problems concerning vibration isolation, transmissibility can be alternatively defined in terms of absolute displacement. Here, absolute displacement transmissibility curves were estimated as well, in order to investigate if there were differences with the absolute acceleration transmissibility representation. It was observed that absolute displacement transmissibility curves differ with respect to the absolute acceleration transmissibility curves in the semi active cases (continuous and ON-OFF). In fact, only in case of linear passive control, acceleration and displacement transmissibility are equal. Therefore, the performance of a semi active control system may result different if absolute acceleration or displacement transmissibility are alternately considered. Besides, the effectiveness of a control strategy should be always checked with the absolute acceleration transmissibility in case of acceleration sensitive equipment.

Equipment relative displacement was checked by evaluating two other quantities. The maximum displacement amplitude was limited across the whole frequency spectrum only with semi active continuous control. Instead concerning the displacement ratio, only light differences were observed comparing the curves obtained with semi active continuous and ON-OFF control, whereas significantly differences are evident with respect to passive control, especially in correspondence to the resonance condition.

Since the input action and its frequency content is not considered always known, it seems difficult to optimize

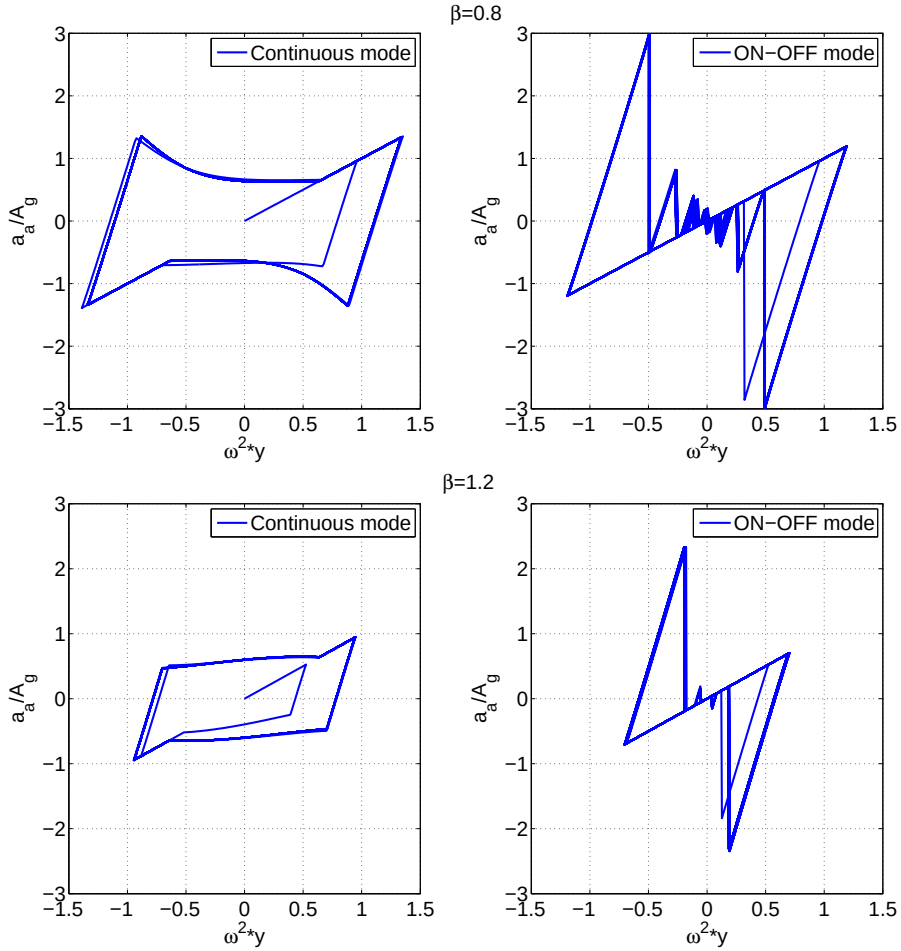

FIG. 5.6. *Absolute acceleration-displacement cycles for semi active continuous (ρ=0.7, λ=5) and ON-OFF (ρ=0, λ=5) control for β=0.8, 1.2.*

the algorithm parameter with the frequency ratio. However, if the expected input action has a frequency content mainly in the region over β=0.6, the solution with ρ=0.7 should be preferred. For equipment acceleration sensitive, the continuous control law gives superior performances with respect to the others: acceleration is well limited and displacement is controlled as well.

In the optimization process, a discontinuity has been observed in the choice of the optimal algorithm and device parameters in the frequency ratio zone around β=0.6. Such region deserves a deeper investigation in a future work by the authors in order to explore the peculiar dynamic behavior of the control system.

REFERENCES

[1] E. RENZI AND M. DE ANGELIS, *Optimal semi active control and non linear dynamic response of variable stiffness structures*, Journal of Vibration and Control, 11-10 (2005), pp. 1253–1289.

[2] M. BASILI AND M. DE ANGELIS, *Shaking table experimentation on adjacent structures controlled by passive and semi-active MR dampers*, Journal of Sound and Vibration, 332-13 (2013) pp. 3113–3133.

[3] Y. C. FAN, C. H. LOH, J. N. YANG AND P. Y. LIN, *Experimental performance evaluation of an equipment isolation using MR dampers*, Earthquake Engineering and Structural Dynamics, 38 (2009) pp. 285–305.

[4] L. Y. LU AND G. L. LIN, *Predictive control of smart isolation system for precision equipment subjected to near-fault earthquakes*, Engineering Structures, 30 (2008) pp. 3045–3064.

[5] H. P. GAVIN AND A. ZAICENCO, *Performance and reliability of semi-active equipment isolation*, Journal of Sound and Vibra-

tion, 306 (2007) pp. 74–90.
[6] E. Renzi and M. De Angelis, *Semi active continuous control of base excited structures: an exploratory study*, Journal of Structural Control and Health Monitoring, 17 (2010), pp. 563–589.
[7] L. Y. Lu, G. L. Lin and T. C. Kuo, *Stiffness controllable isolation system for near-fault seismic isolation*, Engineering Structures, 30 (2008) pp. 747–765.
[8] L. Y. Lu and G. L. Lin, *Improvement of near-fault seismic isolation using a resettable variable stiffness damper*, Engineering Structures, 31 (2009) pp. 2097–2114.
[9] M. Basili and M. De Angelis, *Equipment isolation systems by means of semi active control devices*, Proc. Copech at Wetice, IEEE Conference, Parma, Italy, (2014) pp. 237–242.
[10] M. Ahmadian, *On the isolation properties of semiactive dampers*, Journal of Vibration and Control, 5-2 (1999) pp. 217–232.
[11] Y. Liu, T. P. Waters and M. J. Brennan, *A comparison of semi-active damping control strategies for vibration isolation of harmonic disturbances*, Journal of Sound and Vibration, 280-1 (2005) pp. 21–39.
[12] J. A Liu, G. Leitmann and J. M. Kelly, *Single degree of freedom non-linear homogeneous systems*, ASCE Journal of Engineering Mechanics, 120-7 (1994) pp. 1543–1562.

Edited by: Giacomo Cabri
Received: September 15, 2014
Accepted: January 5, 2015

DOI 10.12694/scpe.v15i4.1055
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 15, Number 4, pp. 345–355. http://www.scpe.org

OPTIMIZING CLOUD RESOURCES ALLOCATION FOR AN INTERNET OF THINGS ARCHITECTURE*

BOGDAN MANAȚE† TEODOR-FLORIN FORTIȘ‡ AND VIOREL NEGRU§

Abstract. The optimization of the cloud resources used to power a multi-agent Internet of Things architecture is an important issue which has an important impact on the overall operation cost of the architecture. The resources tenancy is a costly operation, thus their allocation and management should be optimized based on the usage patterns. The infrastructure for the multi-agent system should not be affected by the deployment or maintenance life cycle, operations require parts of the system, or even the entire system to be offline during the execution of scheduled procedures. This paper outlines of the importance of the infrastructure audit, which offers a good insight of how the resources are used, the geographical areas which are heavily used and where the allocation or release of used resources is mandatory. Also, the security audit, in a distributed multi-agent architecture that handles a large number of heterogeneous devices, represents a good mechanism for performance improvement.

Key words: Internet of Things, cloud computing, multi-agent systems

1. Introduction. The recent advances in Internet of Things (IoT) technologies and the cloud computing adoption have led to an increased usage of this two paradigms to create solid architectures that are able to handle hundred of thousands of concurrent connections, at the same time offering a good Quality of Service (QoS) and Quality of Experience (QoE) for the end user. The resources renting from public cloud providers for supporting the backbone infrastructure for an IoT architecture is beneficial for small and medium-sized enterprises (SME) or academic institutions which are trying to reach a large number of clients, because there is no need to upgrade or maintain the physical infrastructure. Even though the cloud client is not completely aware of the exact location of the hardware that delivers the required information, there are methods available that can determine the best route (considering both geographical location and bandwidth) for optimal performance [39].

For handling a big number of connections from a wide range of devices, an Internet of Things architecture should employ a fast, secure, reliable and fail safe infrastructure for the services offered to the end users which rely upon the manner in which the information is collected. Because of different usage patterns that result from the daily routine of different groups of users, the best solution for building an infrastructure for the IoT framework relies on the elasticity provided by the cloud computing paradigm [11, 15, 23, 33]. Also, the operational cost for maintaining the architecture up and running can be significantly reduced by releasing unused resources at daily time intervals when the audit operations report a low usage of the infrastructure.

From the reliability and operational standpoint, the core infrastructure for the multi-agent system needs to bypass any bottlenecks introduced by the on demand created infrastructure. Therefore, any replacement or restart of the virtual machines should be scheduled on isolated groups of virtual machines without affecting the system's response time.

Usually, on a non-complex application, the application access point and the database server are hosted on the same physical server, thus the business logic is very easy to manage. Therefore, the application access point has direct access to the required information without searching and querying any other network nodes that can be scattered in different data centers. In an Internet of Things framework, the business logic is separated on multiple physical servers, because it needs to process and store a large amount of information sent by the clients. Considering the tremendous amount of data that needs to be stored, the database instances are separated on multiple virtual machines, so that the database instances can be grouped in clusters for a better management and distribution of stored data. Given the heterogeneous nature of the information handled by the Internet of Things architecture the data that needs to be stored can be also grouped based on its type, therefore various

*EXTENDED VERSION OF [18]

†West University of Timişoara, Faculty of Mathematics and Informatics, Computer Science Department, bvd. V. Pârvan, 4, 300223,Timişoara, Romania(bogdan.manate@info.uvt.ro).

‡Institute e-Austria Timişoara & West University of Timişoara, Faculty of Mathematics and Informatics, Computer Science Department, bvd. V. Pârvan, 4, 300223,Timişoara, Romania(fortis@info.uvt.ro).

§Institute e-Austria Timişoara & West University of Timişoara, Faculty of Mathematics and Informatics, Computer Science Department, bvd. V. Pârvan, 4, 300223,Timişoara, Romania(vnegru@info.uvt.ro).

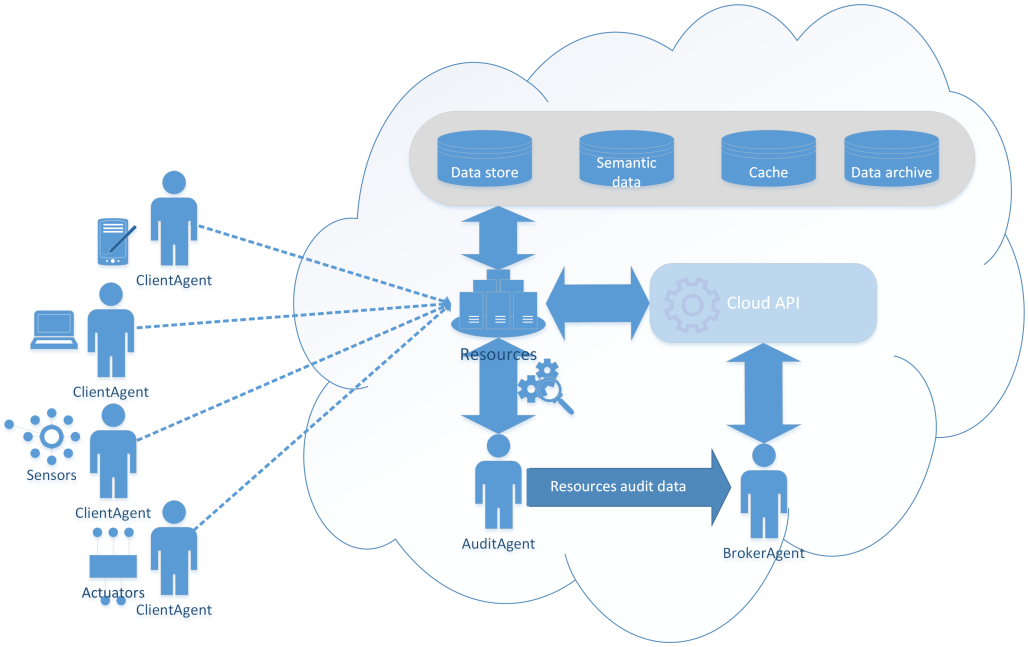

FIG. 1.1. *The actors that interact with the multi-agent architecture [18].*

types of storage solutions can be used based on the information type. The information which is collected on a regular basis from the sensors can be stored in a time series database, the semantic information about devices can be stored using a triple store database and the information about the system's users (e.g. credentials and preferences) can be stored in a document based database.

The same need for intensive infrastructure management is identified in social networks like Facebook [8], Twitter [12], LinkedIn [28] and MySpace [7] and online video streaming providers like Netflix [41] and Hulu [40], where data integrity and services up time play an important role because the services downtime may generate significant revenue losses.

This papers presents how an on demand infrastructure is created to support a multi-agent architecture which operates in the Internet of Things context and how the resources allocation can be optimized based on the usage patterns inferred from the audit operations.

This paper is structured as follows: in Section 2 is presented a literature review of existing infrastructures, in Section 3 is presented the multi-agent architecture designed for Internet of Things governance that needs to manage the underlying cloud infrastructure, in Section 4 is presented a solution for a cloud infrastructure management able to scale an IoT architecture, in Section 5 it is emphasized the importance of infrastructure audit and underline the benefits of a proper infrastructure audit, in Section 6 are presented two methods that enhance the resources utilization and the system scalability and finally, in Section 7 are presented the final conclusions of this paper.

2. Background. By unifying the Internet of Things, Cloud Computing and network edge services a new paradigm emerges called Fog Computing [4], which offers compute power, support for data storage and provides necessary infrastructure for connecting the sensors and clouds. The researchers from CISCO have proposed the Fog Computing term to exemplify a system that is characterize by low latency, ample geographical distribution, location awareness, mobility and heterogeneity. In the context of Fog Computing the services are closer to the consumer delivering this way the required information in a fast and reliable manner. The support for bi-directional data flow (from devices to the cloud and conversely) allows the system to have a greater control over the data sources and the cloud resources. In case of a data source failure, the system can isolate the affected data sources and any important information is inferred from the neighboring devices based on the localization service. Also, the bi-directional data flow enables the multi-agent system to send commands to devices that are

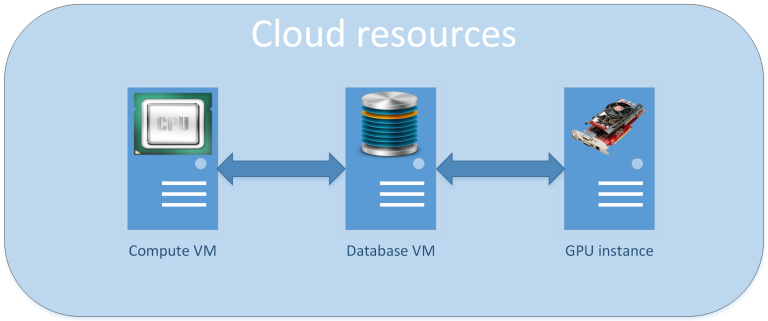

FIG. 2.1. *The types of virtual machines used in the IoT infrastructure [18].*

able to execute simple or complex operations (like actuators).

Beside moving some of the logic to the edge of the network it is imperative to take into account the geographical distribution of the resources that need these services [22]. The services distribution depends on the audit performed on that parts of the network characterized by same location and the number of clients that the system should service. The services are distributed based on the number of possible clients in a given geographical area, therefore a big number of resources will be available for urban areas to service the existing clients and a smaller number of resources will be distributed for the users living in the vicinity of the urban areas. In the edge computing (EC) an important role is played by context awareness, so that a resource it is aware of its current location, the surrounding resources and where is the case the resource can be aware of the entire surrounding context. Thus the methods for information aggregation can benefit from the informations exchanged by the edge services about the surrounding context.

Even though the services are brought as close as possible to the end-users, parts of the data requested by the users is sometimes stored on different database servers situated in various data centers. In [16] is proposed a solution based on the data replication mechanism provided by distributed databases. The important data sets stored in the main data center are replicated on cheap commodity hardware situated at the edge of the network closer to the edge services, reducing by this way the network latency and core network utilization. Also, some of the information can be stored for a small amount of time on the client devices, enabling this way a peer-to-peer information exchange between different users.

By utilizing the multi-agent system based on a distributed infrastructure the end-users should benefit from a good quality of service (QoS) enhanced by a pleasant experience which respects high standards of quality of experience (QoE) [24]. The QoE term is usually used to express the overall experience of using a multimedia system. However, considering the multitude of devices that are encompassed by the Internet of Things framework it is worth noting that these devices are used to improve the user's experience in an environment.

In order to scale the multi-agent architecture proposed in [19], a large number of virtual machine instances are launched in the cloud so that the system load can be evenly distributed on the available resources. The system can be scaled as long as the cloud provider has the necessary resources available, otherwise additional resources may be launched using another public cloud provider or a private cloud. A solution for this problem is offered by the open-source platform mOSAIC [27], which offers an unified application programming interface and a platform for developing large scale applications that are using resources rented from multiple cloud providers. The resources provisioning mechanism is based on a multi-agent architecture (CloudAgency) [38], that enables the platform to rent the best cost effective heterogeneous resources from different cloud providers based on a service level agreement (SLA) provided by the user. The best solution is matched by a semantic reasoning module embeded in the multi-agent system that operates on a Cloud Ontology.

Aneka [37] is another a cloud computing platform that uses cloud resources from the Microsoft Azure cloud for creating services oriented applications. The Aneka platform offers a configurable services container, services discovery and load balancing, therefore the developers can focus on the application development and less on the infrastructure management.

Given the fact that it is impossible for a single cloud provider to scatter its data centers around the globe to

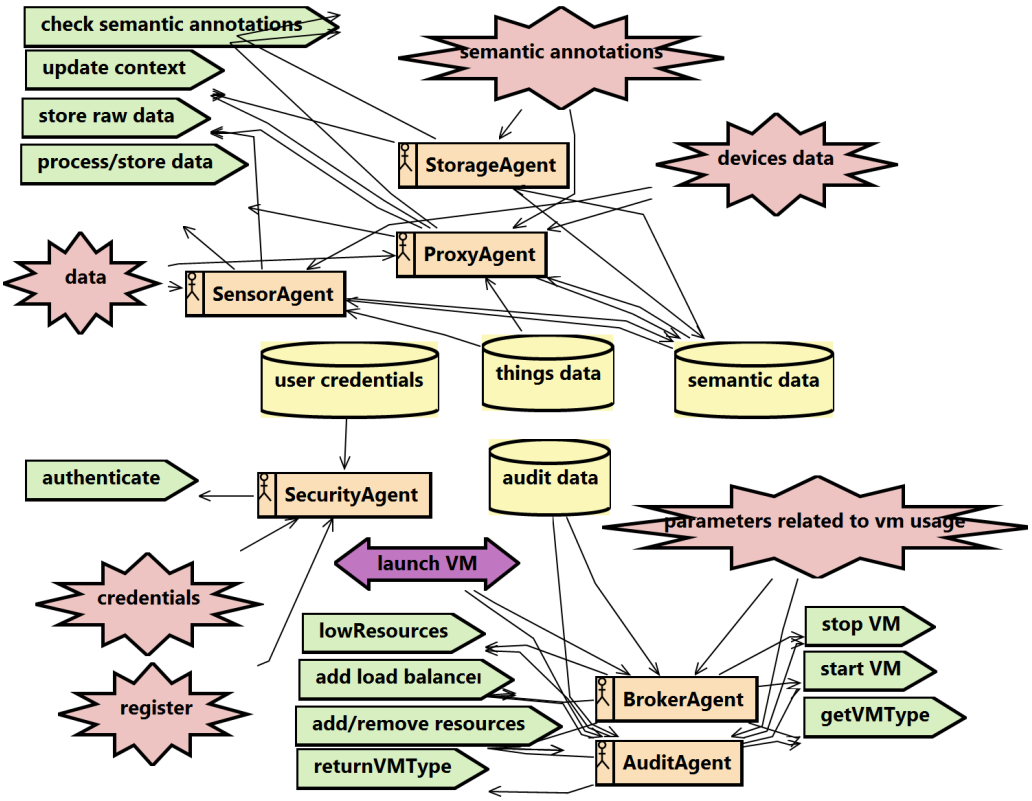

FIG. 3.1. *The multi-agent architecture [19]*

offer on demand computing power as close as possible to their clients, a viable solution to overcome this problem is represented by a cloud federation [5]. By using a federation of clouds and a location discovery services, the resources are rented from the closest available cloud provider thus enabling the infrastructure to offer fast and localized support.

3. Architecture. The proposed architecture for the multi-agent system [19] presented in Figure 3.1 uses the agents as independent and mobile entities, which are specialized to solve specific domain oriented issues that are introduced by the Internet of Things paradigm. As seen in Fig. 1.1, groups of agents are working together to gather, annotate, process and store data in a cloud environment. Other collections of specialized agents are distributed on client devices based on the devices semantic definition. By having the agents distributed both on the client devices and on the cloud infrastructure, the communication and the management of the agents is delegated to the core implementation of the multi-agent platform, thus the resulting system operates as an integrated environment.

The information, which may be sent as partially processed or raw data, flows from the clients to the cloud endpoints that are managed by the instances of the ProxyAgent, which are routing the information based on its type to the available resources. When the information reaches one of the end-points of the cloud, it can be semantically annotated and sent for further processing. Considering the special cases when the context aware component is active, the information related to a particular context can be used for updating the current context if the information received contains relevant values that depict a modification of the actual state. The information that contributes to context modelling is managed by ContexManagerAgent [21], therefore the ProxyAgent needs to route the information related to the context to a virtual machine where the ContextManagerAgent is instantiated.

The data cache plays an important role in a system that handles massive quantities of information because it has a big impact on the system performance. The most accessed information is stored in the RAM memory,

so that all requests trying to read the information from a database are routed directly to the information stored by the cache system. The agents that are handling a large amount of information are developed with an of the shelf local caching system to reduce the time needed for information access.

When data gathered from the devices is older than a specified amount of time it is transferred to the data archive database. The archive database compress and stores the aggregated information in order to facilitate a later information retrieval of the archived data. When the information stored in the archive database becomes obsolete the data archive is subjected to a purge operation that aims to remove the existing information which has not been used in a long time.

The multi-agent system was developed to serve a general purpose when it comes to data collection and processing, but it can be used in various domains like ambient intelligence, smart city management, ambient assisted living and for supervising the industrial processes.

4. Infrastructure management. The proposed multi-agent system employs three types of virtual machines, which have to accomplish a specific task in the architecture:

- Compute VM - compute optimized instances which offer the highest performing processors. These types of instances are used for the data collection end-points. The multi-agent container runs on these instances, where agents specialized for data processing and semantic annotation are instantiated.
- Database VM - storage optimized instances that are able to offer support for memory-intensive and random I/O operations. The instances of this type can be grouped into clusters depending on the technical specifications provided by the company which delivers the database software solution. A virtualization advisor [34] can be used to determine based on the audit data which is the best configuration for the database instances.
- GPU instances - instances that have attached a graphical and general purpose GPU. This type of instance is used for data mining [17], data-matching [29], intrusion detection [14, 36] and learning [10].

4.1. Managing compute instances. In the paper [20] it is presented a suite of benchmark tests performed on a prototypical implementation of the Internet of Things architecture briefly presented in Section 3. The benchmarks results define the maximum number of connections that can be handled by different types of virtual machines rented from Amazon EC2. The maximum threshold for every virtual machine type can be used by the BrokerAgent, as a trigger, to launch a new instance in order to distribute the system load.

The standard virtual machine images, that are built with the purpose of supporting the architecture, are bundled with necessary software packages to start the architecture. When a non standard image is launched from the cloud provider repository, the BrokerAgent handles the deployment phase as well the registering phase after which the image is marked as *ready* to process the data collected from the devices.

Event though the implementation of the proposed multi-agent architecture targets the Scala programming language and Akka toolkit, to benefit from the highly concurrent and distributed nature of the proposed multi-agent system, the out of the box scheduling mechanism [3] implemented by Scala/Akka software stack can be greatly improved with the addition of load balancers [32], which are offered by default by the majority of the cloud providers. Therefore, the BrokerAgent is designed to handle this scenario by grouping the targeted virtual machines, automatically, requesting a new load balancer for the grouped instances from the cloud provider using the cloud provider's public API.

Another benefit resulted from the usage of a multi-agent system on top of the Akka toolkit is the standard implementation of the communication protocol, which is the default option supported by every agent. This way the agents are able to exchange information regardless of their location, hence, the infrastructure might be exposed to a higher network traffic when the actors are deployed on a virtual machine that has been launched in a different data centers.

The audit operations performed by the AuditAgent on the entire infrastructure are important, because they offer valuable information about the resources utilization, unauthorized access attempts, system loading, the number of newly added/removed devices and the geographical distribution of the end users. In order to offer edge computing services for the clients the system should be aware of the resources' positions which are useful when a local system failure occurs. By knowing the exact location of the resources and the location of the clients, the edge services can be dynamically managed and deployed offering a flexible alternative when a local resource becomes unavailable.

4.2. Managing database instances. The instance type that is hosting the database software solutions represents an important instance type for the architecture, because it assures the data persistence. The database instances need to be configured based on the technical specifications of the database software which is launched on these instances. The special situation that arise when database instances are grouped together in a cluster or ring require additional instructions to join the group after a restart. Also, other tuning operations can be executed to improve the database response time. Therefore, the BrokerAgent has to mitigate the interaction between the cluster and the new database instance and any other operations that require interaction with the database software. For applications targeting the Java platform, the JSch [1] library can be used to send commands over SSH to a virtual machine in a secure manner.

The operations executed via the command line interpreter, like joining a cluster, gathering audit data or a force data replication, are executed by any of the agents running on the local agents container or running on a remote virtual machine, thus allowing the multi-agent system to have total control over the launched instances. Therefore, for security reasons, the interactions with the underlying operating system should be executed only by the agents which have a very well defined role in the infrastructure management like AuditAgent or BrokerAgent.

Some NoSQL databases, like Riak, recommend the usage of a load balancer [1] as a best practice, because every node that is a member of a cluster is able to handle the incoming requests, hence the incoming requests can be routed to any available instance.

4.3. Managing GPU instances. The GPU instances are offered at high rates, because the GPU boards are bundled with expensive hardware. Hence, only a handful of cloud providers offer such instances, some of them are big players in cloud computing domain like Amazon, Nimbix, Peer1, Penguin computing and Softlayer.

The data processing using a high performance GPU is very fast, hence the data traffic between the GPU instances and the data store can simply overcome the infrastructure network capability. To solve this problem, the GPU instances need to be started in the same data center where the data which is subject to processing is available. Also, the network connection between the database instances and the GPU instances should offers support for a higher bandwidth to speed up the data transfer. A better solution for networking is a cluster network where the instances launched in the same cluster group are started on the same physical server rack, so that the cluster network provides high network bandwidth and low latency for data transfer between instances.

The agents of the proposed multi-agent system that are running tasks on the GPU instances are implemented using ScalaCL [2] a library that lets the programmers to run Scala code on the GPUs in a very natural way. The tasks distribution is handled by a router agent, which uses a Round Robing algorithm to distribute the information stored in the local mailbox (an internal queue).

5. Infrastructure audit. The system audit is important in an environment where tens of thousands of virtual machines represent the backbone of a multi-agent architecture (Fig. 5.1). The audit is useful for automatic maintenance as well as for human operators, so that the data gathered by the audit operation can be used to maximize the infrastructure performance and to reduce the operational costs. The audit operations offer important data about the network latency, CPU usage, GPU usage, RAM memory loading, and it can also verify if the resources rented from the cloud provider respect the service level agreement (SLA) [25], which is very important for the infrastructure stability. The multi-agent system relies on the cloud infrastructure to operate at best performance parameters, so that any change in the infrastructure components can affect the system's overall performance. Thus, the audit operations must be scheduled after every start/restart of a virtual machine to validate the changes [6].

Because the multi-agent architecture proposed in [19] was developed to target a wide-spread geographical area, the audit information is useful to determine which area offers a high QoS/QoE for end-users. Thus, the edge computing services offered for certain geographical areas can be configured for best performance results considering the audit data collected for the specified areas.

The AuditAgent uses a database to store different audit results based on the execution time. Storing audit results for a long period of time is useful to identify the parts of the infrastructure that are extensively used so

[1] http://www.jcraft.com/jsch/
[2] https://code.google.com/p/scalacl/

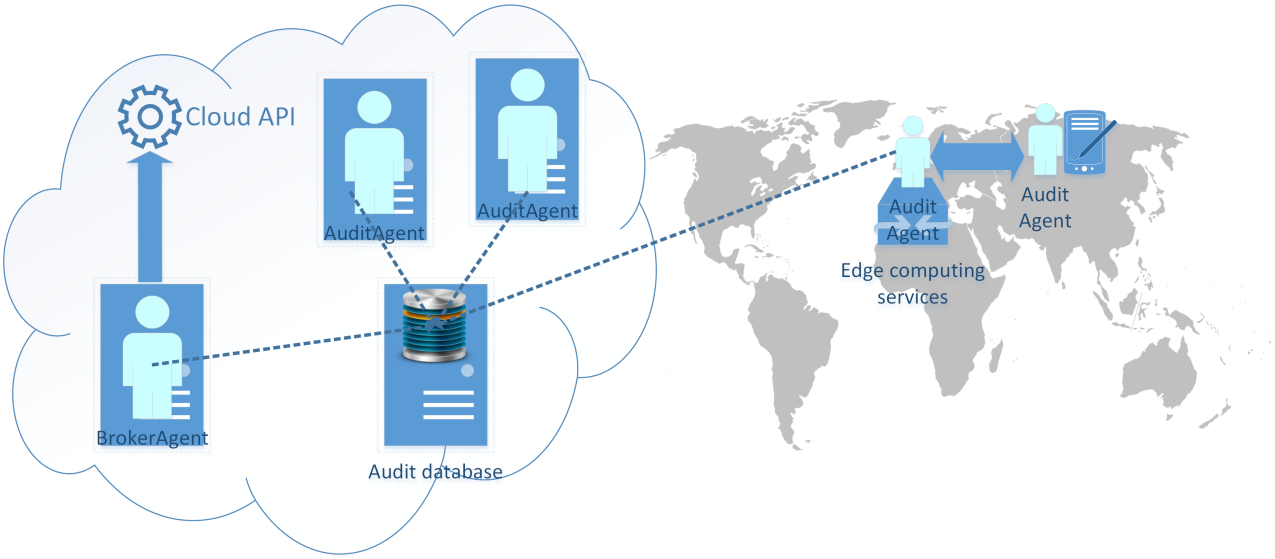

FIG. 5.1. *The IoT infrastructure audit [18].*

that further actions may be taken to improve performances by moving some of the resources situated in areas with low traffic to the area where the infrastructure is experiencing peaks of traffic.

Considering the vast applications of the Internet of Things in domains like health care, smart city, ambient intelligence and industrial, a lot of private information is transferred between agents which are situated either on the client side or in the cloud. To protect the private information of the end-users it is imperative to execute security audit operations [35] over different components of the system. The heterogeneous character of the devices that interact in the IoT framework offer a wide range of possibilities for a cyber-attack, therefore periodic audit operations are compulsory. The security audit needs to target the client-side applications, edge computing services and the cloud infrastructure periodically, so that any attempt or unauthorized usage of the system has to be immediately identified and reported.

Another important aspect of the infrastructure audit is related to the detection of the intrusion attempts. The detection of intrusion attempts can be automatically handled by the SecurityAgent, which has a set of rules implemented to deal with such situations. Real time intrusion detection for the Internet of Things paradigm has been implemented in SVELTE [31] project with an impressive detection rate of almost 100%, thus the task of detecting potential attackers can be assigned to an agent which has full access to the data exchange inside of the multi-agent system.

6. Resources usage optimization. Even though there are many research papers [2, 9, 13, 26, 30] focused on the optimization of the cloud resources usage, for the particular case regarding the proposed multi-agent architecture the resource usage can be optimized using a two-phase method base on resources pooling and virtual machine pre-warm-up. In order to optimize the resources usage in an Internet of Things architecture there should be a balance between assuring the system stability and resources usage. The first method presented bellow is based on a pool of resources and is employed when no information about the system usage is available. After the audit operations manage to gather a significant amount of data about how and when the system is accessed by the end users the multi-agent architecture will be able to pre-warm the virtual machines just in time for the expected traffic peak.

6.1. Resources pooling. The resource pooling method is used when there is no audit data available about the system's usage so that the virtual machines cannot be prepared for the network traffic peaks. A variable number of compute virtual machines are provisioned with the required software stack and are kept in a resources pool for when the system is under heavy load. As seen in Fig. 6.1 when the system is under heavy load the available resources are moved from the resources pool and attached to the load balancer that is

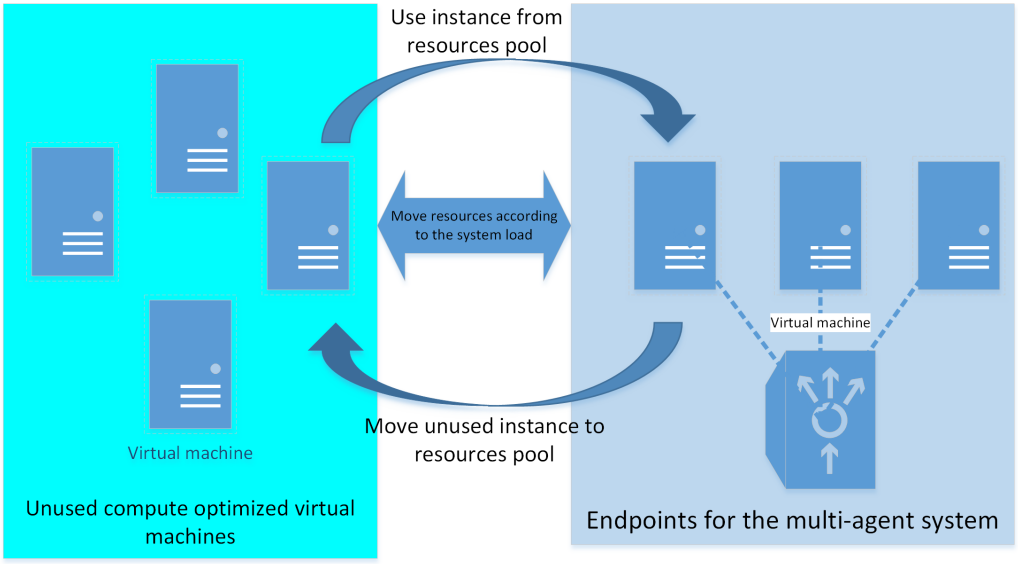

FIG. 6.1. *Resources pool.*

experiencing heavy network traffic.

Keeping the underutilized resources running is the main drawback of this method but is a fair compromise for keeping the system running in special situations generated by massive network traffic.

6.2. Virtual machines 'warm-up'. The Internet of Things is characterized by highly localized and repetitive events, therefore after collecting enough information about the system usage it is very easy to forecast where and when are required extra cloud resources. The information about the system usage can be extracted by analyzing the audit data collected by the AuditAgent. The number of virtual machines present in the resources pool can be gradually decreased as the information gathered from the audit operations has relevant information about the usage time frame.

In Fig. 6.2 is presented a general usage sample of the available resources. The point B represents the maximum number of connections that can be handled by the multi-agent system using the existing resources. The point C represents a forecast of the maximum number of connections for the current time interval. In order to avoid an overloading of the system by reaching the critical point B, when no additional virtual machines are started to take over the excess traffic, the point A was selected as the best time when a new virtual machine should be started. The best time is calculated by subtracting the time needed for the virtual machine to start (the time needed for booting and for installing the required software stack) from the time when the critical point B will be reached.

7. Conclusion and future work. This paper presents a method for a cloud infrastructure management and cloud resources allocation based on a multi-agent solution. The proposed solutions aim to overcome the issues encountered when managing an on-demand cloud infrastructure for such a dynamic framework like the Internet of Things. Because the entire logic for infrastructure management is detached to specialized agents, that operate on data collected from the system logs or from user defined constraints, it can be used as a standalone component in other contexts, that have similar requirements like the multi-agent system used for Internet of Things governance and which are required to operate using the infrastructure as a service paradigm.

Another important aspect underlined in this paper is related to infrastructure audit and the positive impact on the system performance when the data collected during the audit operation is used to dynamically reconfigure the system. As presented in Section 6 the audit operation also play an important role for resources usage optimization. The audit information is useful when dealing with location aware services, because it offers valuable information about the current state of the machines which are hosting the edge network services.

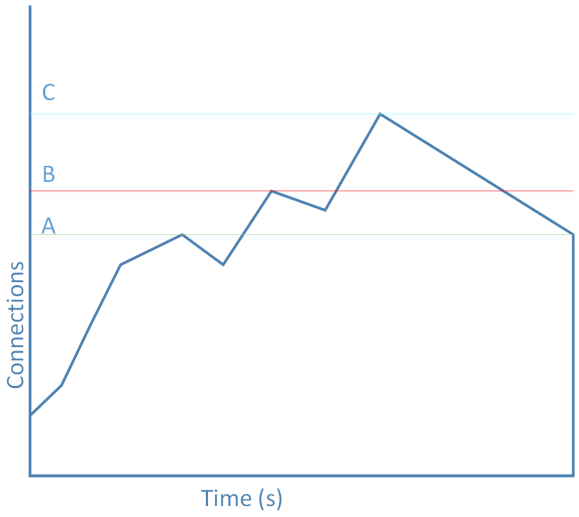

FIG. 6.2. *A general usage sample*

Hence, the regions which are characterized by heavy network traffic can benefit from a new set of computing resources during traffic peak.

For further research it is planned to develop a feature of the multi-agent system that analyses the instances usage patterns related to users profiles, so that the agents will be able to significantly reduce the number of the instances launched in the resources pool. Therefore the multi-agent system will provide just in time the instances needed for balancing the incoming requests, thus reducing the unnecessary costs generated by the idle instances from the resources pool.

Acknowledgment. The work of the first author was partially supported by the strategic grant POSDRU/159/1.5/S/137750, "Project Doctoral and Postdoctoral programs support for increased competitiveness in Exact Sciences research" cofinanced by the European Social Fund within the Sectoral Operational Programme Human Resources Development 2007–2013 and Romanian Government grant PN-II-ID-PCE-2011-3-0260 (AMICAS). The views expressed in this paper do not necessarily reflect those of the corresponding projects consortium members.

REFERENCES

[1] *Riak load balancing.* `http://docs.basho.com/riak/1.3.1/cookbooks/Load-Balancing-and-Proxy-Configuration/`. Accessed: 2014-09-08.

[2] V. AGGARWAL, V. GOPALAKRISHNAN, R. JANA, K. RAMAKRISHNAN, AND V. A. VAISHAMPAYAN, *Optimizing cloud resources for delivering IPTV services through virtualization*, in Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on, IEEE, 2012, pp. 1–10.

[3] M. BEVILACQUA-LINN, M. BYRON, P. CLINE, J. MOORE, AND S. MUIR, *Sirius: distributing and coordinating application reference data*, in Proceedings of the 2014 USENIX conference on USENIX Annual Technical Conference, USENIX Association, 2014, pp. 293–304.

[4] F. BONOMI, R. MILITO, J. ZHU, AND S. ADDEPALLI, *Fog computing and its role in the internet of things*, in Proceedings of the first edition of the MCC workshop on Mobile cloud computing, ACM, 2012, pp. 13–16.

[5] R. BUYYA, R. RANJAN, AND R. N. CALHEIROS, *Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services*, in Algorithms and architectures for parallel processing, Springer, 2010, pp. 13–31.

[6] F. DOELITZSCHER, C. FISCHER, D. MOSKAL, C. REICH, M. KNAHL, AND N. CLARKE, *Validating cloud infrastructure changes by cloud audits*, in Services (SERVICES), 2012 IEEE Eighth World Congress on, IEEE, 2012, pp. 377–384.

[7] M. FARKAS, *Going where patrons are: Outreach in MySpace and Facebook*, American Libraries, 38 (2007), p. 27.

[8] N. FARRINGTON AND A. ANDREYEV, *Facebooks Data Center Network Architecture*, in IEEE Opt. Interconnects Conf, Citeseer, 2013, pp. 5–7.

[9] J. M. FERRIS, *Methods and systems for optimizing resource usage for cloud-based networks*, Aug. 22 2008. US Patent App. 12/196,459.

[10] L. F. Gruber and M. West, *GPU-Accelerated Bayesian Learning and Forecasting in Simultaneous Graphical Dynamic Linear Models*, tech. report, Technical report, Department of Statistical Science, Duke University, 2014.
[11] D. Guinard, C. Floerkemeier, and S. Sarma, *Cloud computing, REST and mashups to simplify RFID application development and deployment*, in Proceedings of the Second International Workshop on Web of Things, ACM, 2011, p. 9.
[12] T. Hoff, *Scaling Twitter: Making Twitter 10000 Percent Faster*, High Scalability, (2009).
[13] S. Khatua, A. Ghosh, and N. Mukherjee, *Optimizing the utilization of virtual resources in Cloud environment*, in Virtual Environments Human-Computer Interfaces and Measurement Systems (VECIMS), 2010 IEEE International Conference on, IEEE, 2010, pp. 82–87.
[14] S.-i. Kim, W. Edmonds, and N. Nwanze, *On GPU accelerated tuning for a payload anomaly-based network intrusion detection scheme*, in Proceedings of the 9th Annual Cyber and Information Security Research Conference, ACM, 2014, pp. 1–4.
[15] M. Kovatsch, S. Mayer, and B. Ostermaier, *Moving application logic from the firmware to the cloud: Towards the thin server architecture for the internet of things*, in Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on, IEEE, 2012, pp. 751–756.
[16] Y. Lin, B. Kemme, M. Patino-Martinez, and R. Jimenez-Peris, *Enhancing edge computing with database replication*, in Reliable Distributed Systems, 2007. SRDS 2007. 26th IEEE International Symposium on, IEEE, 2007, pp. 45–54.
[17] W. Ma and G. Agrawal, *A translation system for enabling data mining applications on GPUs*, in Proceedings of the 23rd international conference on Supercomputing, ACM, 2009, pp. 400–409.
[18] B. Manaţe, T.-F. Fortiş, and V. Negru, *Infrastructure Management Support in a Multi-Agent Architecture for Internet of Things*, in European Modelling Symposium EMS2014 (EMS2014), Pisa, Italy, Oct. 2014.
[19] B. Manaţe, T.-F. Fortis, and P. Moore, *Applying the Prometheus methodology for an Internet of Things architecture*, in Utility and Cloud Computing (UCC), 2014 Fourth IEEE International Conference on Utility and Cloud Computing, IEEE, 2014.
[20] B. Manaţe, V. I. Munteanu, and T.-F. Fortiş, *Towards a Scalable Multi-agent Architecture for Managing IoT Data*, in P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference on, 2013, pp. 270–275.
[21] B. Manaţe, V. I. Munteanu, T. F. Fortis, and P. T. Moore, *An intelligent context-aware decision-support system oriented towards healthcare support*, Complex, Intelligent and Software Intensive Systems (CISIS), 2014 Eighth International Conference, (2014), pp. 386–391.
[22] R. Manning, *Dynamic and distributed managed edge computing (MEC) framework*, May 20 2004. US Patent App. 10/850,291.
[23] N. Mitton, S. Papavassiliou, A. Puliafito, and K. S. Trivedi, *Combining Cloud and sensors in a smart city environment*, EURASIP Journal on Wireless Communications and Networking, 2012 (2012), pp. 1–10.
[24] R. F. Moghaddam and M. Cheriet, *A Note on Quality of Experience (QoE) beyond Quality of Service (QoS) as the Baseline*, arXiv preprint arXiv:1407.5527, (2014).
[25] D. Ouelhadj, J. Garibaldi, J. MacLaren, R. Sakellariou, and K. Krishnakumar, *A multi-agent infrastructure and a service level agreement negotiation protocol for robust scheduling in grid computing*, in Advances in Grid Computing-EGC 2005, Springer, 2005, pp. 651–660.
[26] N. Paton, M. A. De Aragão, K. Lee, A. A. Fernandes, and R. Sakellariou, *Optimizing utility in cloud computing through autonomic workload execution*, Bulletin of the Technical Committee on Data Engineering, 32 (2009), pp. 51–58.
[27] D. Petcu, B. Di Martino, S. Venticinque, M. Rak, T. Máhr, G. E. Lopez, F. Brito, R. Cossu, M. Stopar, S. Šperka, et al., *Experiences in building a mOSAIC of clouds*, Journal of Cloud Computing, 2 (2013), pp. 1–22.
[28] J. M. Pujol, G. Siganos, V. Erramilli, and P. Rodriguez, *Scaling online social networks without pains*, in Proc of NETDB, 2009.
[29] C.-P. Pungila, M. Reja, and V. Negru, *Efficient parallel automata construction for hybrid resource-impelled data-matching*, Future Generation Computer Systems, 36 (2014), pp. 31–41.
[30] J. Rao, X. Bu, C.-Z. Xu, and K. Wang, *A distributed self-learning approach for elastic provisioning of virtualized cloud resources*, in Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on, IEEE, 2011, pp. 45–54.
[31] S. Raza, L. Wallgren, and T. Voigt, *SVELTE: Real-time intrusion detection in the Internet of Things*, Ad hoc networks, 11 (2013), pp. 2661–2674.
[32] M. Sharma, Y. Anitha, and P. Sharma, *An Optimistic Approach for Load Balancing in Cloud Computing*, (2014).
[33] J. Soldatos, M. Serrano, and M. Hauswirth, *Convergence of utility computing with the internet-of-things*, in Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on, IEEE, 2012, pp. 874–879.
[34] A. A. Soror, U. F. Minhas, A. Aboulnaga, K. Salem, P. Kokosielis, and S. Kamath, *Automatic virtual machine configuration for database workloads*, ACM Transactions on Database Systems (TODS), 35 (2010), p. 7.
[35] X. Sun and C. Wang, *The Research of Security Technology in the Internet of Things*, in Advances in Computer Science, Intelligent System and Environment, Springer, 2011, pp. 113–119.
[36] G. Vasiliadis, S. Antonatos, M. Polychronakis, E. P. Markatos, and S. Ioannidis, *Gnort: High performance network intrusion detection using graphics processors*, in Recent Advances in Intrusion Detection, Springer, 2008, pp. 116–134.
[37] C. Vecchiola, X. Chu, and R. Buyya, *Aneka: a software platform for .NET-based cloud computing*, High Speed and Large Scale Scientific Computing, (2009), pp. 267–295.
[38] S. Venticinque, R. Aversa, B. Di Martino, and D. Petcu, *Agent based Cloud Provisioning and Management-Design and Prototypal Implementation.*, in CLOSER, 2011, pp. 184–191.
[39] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, *Cloudanalyst: A cloudsim-based visual modeller for analysing cloud*

computing environments and applications, in Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on, IEEE, 2010, pp. 446–452.

[40] F. Wu, W.-H. Chen, P. R. Graham, and G. D. Pelton, *Efficiently distributing video using a hybrid network that uses existing infrastructure*, Oct. 13 2009. US Patent 7,602,846.

[41] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, *Large-scale parallel collaborative filtering for the netflix prize*, in Algorithmic Aspects in Information and Management, Springer, 2008, pp. 337–348.

Edited by: Dana Petcu
Received: November 15, 2014
Accepted: January 15, 2015

AIMS AND SCOPE

The area of scalable computing has matured and reached a point where new issues and trends require a professional forum. SCPE will provide this avenue by publishing original refereed papers that address the present as well as the future of parallel and distributed computing. The journal will focus on algorithm development, implementation and execution on real-world parallel architectures, and application of parallel and distributed computing to the solution of real-life problems. Of particular interest are:

Expressiveness:

- high level languages,
- object oriented techniques,
- compiler technology for parallel computing,
- implementation techniques and their efficiency.

System engineering:

- programming environments,
- debugging tools,
- software libraries.

Performance:

- performance measurement: metrics, evaluation, visualization,
- performance improvement: resource allocation and scheduling, I/O, network throughput.

Applications:

- database,
- control systems,
- embedded systems,
- fault tolerance,
- industrial and business,
- real-time,
- scientific computing,
- visualization.

Future:

- limitations of current approaches,
- engineering trends and their consequences,
- novel parallel architectures.

Taking into account the extremely rapid pace of changes in the field SCPE is committed to fast turnaround of papers and a short publication time of accepted papers.

INSTRUCTIONS FOR CONTRIBUTORS

Proposals of Special Issues should be submitted to the editor-in-chief.

The language of the journal is English. SCPE publishes three categories of papers: overview papers, research papers and short communications. Electronic submissions are preferred. Overview papers and short communications should be submitted to the editor-in-chief. Research papers should be submitted to the editor whose research interests match the subject of the paper most closely. The list of editors' research interests can be found at the journal WWW site (`http://www.scpe.org`). Each paper appropriate to the journal will be refereed by a minimum of two referees.

There is no a priori limit on the length of overview papers. Research papers should be limited to approximately 20 pages, while short communications should not exceed 5 pages. A 50–100 word abstract should be included.

Upon acceptance the authors will be asked to transfer copyright of the article to the publisher. The authors will be required to prepare the text in LaTeX 2$_\varepsilon$ using the journal document class file (based on the SIAM's `siamltex.clo` document class, available at the journal WWW site). Figures must be prepared in encapsulated PostScript and appropriately incorporated into the text. The bibliography should be formatted using the SIAM convention. Detailed instructions for the Authors are available on the SCPE WWW site at `http://www.scpe.org`.

Contributions are accepted for review on the understanding that the same work has not been published and that it is not being considered for publication elsewhere. Technical reports can be submitted. Substantially revised versions of papers published in not easily accessible conference proceedings can also be submitted. The editor-in-chief should be notified at the time of submission and the author is responsible for obtaining the necessary copyright releases for all copyrighted material.