Software & Systems Modeling (2019) 18:2137–2176
https://doi.org/10.1007/s10270-017-0645-2

REGULAR PAPER

Multidimensional context modeling applied to non-functional analysis of software

Luca Berardinelli[1] · Marco Bernardo[2] · Vittorio Cortellessa[3] · Antinisca Di Marco[4]

Received: 14 October 2016 / Revised: 31 October 2017 / Accepted: 13 November 2017 / Published online: 16 December 2017

Abstract
Context awareness is a first-class attribute of today software systems. Indeed, many applications need to be aware of their context in order to adapt their structure and behavior for offering the best quality of service even in case the software and hardware resources are limited. Modeling the context, its evolution, and its influence on the services provided by (possibly resource constrained) applications are becoming primary activities throughout the whole software life cycle, although it is still difficult to capture the multidimensional nature of context. We propose a framework for modeling and reasoning on the context and its evolution along multiple dimensions. Our approach enables (1) the representation of dependencies among heterogeneous context attributes through a formally defined semantics for attribute composition and (2) the stochastic analysis of context evolution. As a result, context can be part of a model-based software development process, and multidimensional context analysis can be used for different purposes, such as non-functional analysis. We demonstrate how certain types of analysis, not feasible with context-agnostic approaches, are enabled in our framework by explicitly representing the interplay between context evolution and non-functional attributes. Such analyses allow the identification of critical aspects or design errors that may not emerge without jointly taking into account multiple context attributes. The framework is shown at work on a case study in the eHealth domain.

Keywords Context modeling · Context evolution · Reliability · Performance · Transient and steady-state analysis

1 Introduction

Software is increasingly pervading our daily life, as ever more tasks that, up to few years ago, were performed by mechanical devices are now delegated to software automation. Beside this, the development of large bandwidth networks has enabled the easy interaction among geographically distributed software components that collaborate to offer new software services to the users. These evolution trends have brought, in the last few years, software applications to be required to run in extremely different contexts without failures and without degradation of their quality.

By *context*, it is here intended the (often heterogeneous) information that a software system is capable of sensing by itself (e.g., through reflection mechanisms [7]) or from the external environment (e.g., through sensors), and that can influence the structure and the behavior of the services provided by the system [29]. This abstract and general definition needs to be instantiated according to the particular sensing capabilities of the applications. Hence, we consider the context as a *heterogeneous* and *application-specific* property.

Communicated by Professor Dorina Petriu.

✉ Luca Berardinelli
luca.berardinelli@tuwien.ac.at

Marco Bernardo
marco.bernardo@uniurb.it

Vittorio Cortellessa
vittorio.cortellessa@univaq.it

Antinisca Di Marco
antinisca.dimarco@univaq.it

[1] Distributed Systems Group, Technische Universität Wien, Vienna, Austria

[2] Dipartimento di Scienze Pure e Applicate, Università di Urbino, Urbino, Italy

[3] Dipartimento di Ingegneria e Scienze dell'Informazione, e Matematica, Università dell'Aquila, L'Aquila, Italy

[4] Dipartimento di Scienze Cliniche Applicate e Biotecnologiche, Università dell'Aquila, L'Aquila, Italy

Context awareness is a prerequisite to enable *adaptation*, which is the ability of an application to change its structure and/or behavior in response to the context evolution [34], aimed at satisfying requirements. *Context evolution* is the change of context information during the system life cycle, from its requirement specification to implementation, deployment, and execution.

In this respect, context and its evolution may affect the quality of service (QoS) in terms of variations of non-functional properties that may result in (hopefully temporary) violations of non-functional requirements (NFR) [16]. In order to avoid such violations, a context-aware application should be able to evaluate the QoS in the current execution context, as well as to predict how the QoS varies with respect to the context evolution.

The aim of this paper is the introduction of a framework for: (1) describing and reasoning on heterogeneous context attributes and their (isolated or combined) evolution, and (2) modeling and analyzing non-functional properties of multi-dimensional context-aware systems.

For the former goal, we introduce a unifying representation of context based on a stochastic extension of statecharts [26], where each state represents (a combination of) context values and the probabilistic transitions represent the context evolution. For the latter goal, we build on the experience gained in model-based methodologies for non-functional analysis of software systems, and we extend them to the domain of context-aware applications.

Beyond what we show in this paper, the context modeling approach we propose has several potential applications, both during the software development process and after software deployment:

- *During the software development process*, it can be exploited to better allocate testing activities in order to focus efforts on those context states where the system stays longer, or it can be used to calibrate (hardware and software) resources in each context state, possibly in favor of those where the system sojourns longer.
  In this paper, we show in Sect. 5 how to apply the proposed context modeling to performance and reliability analysis of software.
- *After software deployment*, it can allow to determine whether the system and the underlying assumptions are valid and conform to the stakeholder desiderata. In particular, it permits to answer questions such as: "Which is the probability that a doctor's device is in low-power mode while the doctor is traveling from the surgery room to the patient's home?". This information can reveal, for example, the inadequacy of the battery of the doctor device. Or "What is the impact of the duration of the doctor's visit to the patients on other activities?". Just to make another example, in the domain of cyber-physical production systems (CPPS), it can help to answer questions such as: "Which is the probability that, while machine A is in its self-maintenance phase, the operator is exactly in the same room?", or "Which is the probability that the human operator is in the factory and not at home while the machine A turns its status in self-maintenance phase?". This kind of prediction helps to understand whether the current operator schedule is acceptable for a specific CPPS.[1]

In this paper, first we provide a quantitative context analysis methodology whose results can be evaluated per se. Second, we show how our approach can be combined with other engineering activities like model-based NFR analyses to answer questions like: "What happens to non-functional attributes if context evolves along a certain trend?", or "Does the designed adaptation strategy permit to guarantee NFR over time?", which cannot be addressed without combining context, design, and non-functional modeling. Our solution raises the level of abstraction at which designers deal with context representation and allows them to capture the interdependencies between system design, non-functional analyses, and context.

This paper is an extended version of [8], where the extensions are: (1) the introduction of a formal composition semantics for context modeling (see Sects. 3.1 and 3.2), (2) the transient-state and sensitivity context analyses (see Sect. 4.2), and (3) the integration of context modeling and analysis with an existing model-based reliability approach (see Sect. 5.2.1).

The paper is organized as follows: Section 2 introduces Mobile eHealth (MeH), which is a context-aware application that will be used as a leading case study throughout the paper. Section 3 illustrates the basic ideas behind the proposed context modeling approach and formalizes the whole approach with particular emphasis on a compositional semantics for context attributes. Section 4 illustrates the context evolution analysis. Section 5 shows how to combine the proposed context model with UML as hosting notation and then with model-based performance and reliability analysis approaches, so that interdependencies among system design, non-functional analyses, and context can be studied. Section 6 presents related work and points out the novelty of this paper with respect to the state of the art. Section 7 concludes the paper and indicates directions for future work.

[1] Our ongoing work in this direction is available at: http://me-at-big.blogspot.co.at/2016/07/context-modeling-and-analysis-of-cyber.html.

2 An illustrative case study: a mobile eHealth application

In this section, we introduce the mobile eHealth application (MeH) that will be used throughout the paper as leading case study.

MeH aims at providing services to support doctors' everyday activities. A critical one is the retrieval of mixed media information on the assisted patients (Request Patient Data service, RPD) that combines descriptive text with different kinds of images referring to patients' personal data, medical histories, and diseases (e.g., X-ray images).

The result of the RPD service is a multimedia report that can be displayed on the doctor's personal digital assistant (PDA). The application client, which is deployed on the PDA, is capable of connecting different communication networks, where available, and choosing the best one (e.g., the one with the highest bandwidth) among the networks available at the current user location.

Long and/or heavy computational tasks, like the download of large images and/or their editing, depend on the status of resources available on the client side, such as a sufficient charge level of the battery. For example, the frequency of the CPU equipping the PDA, as well as the brightness of its display, may be limited to reduce the power consumption when the charge level of the PDA battery is lower than a certain threshold.

In addition, the doctor may invoke the MeH services while moving across different physical places (e.g., at home, at the surgery, at patients' home, or outdoor), where different resources may be available and/or their exploitation may change. For example, a high-bandwidth network and a PDA battery recharger may be available only when the doctor stays indoors, and not in other locations.

Therefore, MeH should collect heterogeneous data from different context sources like users (e.g., the doctor), hardware components (e.g., the battery of the PDA), or the external environment (e.g., the WiFi) in order to provide the best QoS.

3 Context modeling through composable context evolution models

In the literature, several approaches support the idea of modeling physical or logical *location awareness* (referred to as *spatial context models* in [13]) with state machines or similar notations [8,20], where a state represents the current location (of a physical device or a software component) and a transition represents a change of location. A location is usually defined as a (logical or physical) place characterized by the resources accessible in that place. Multiple physical or logical places may have the same resources, and therefore, they can be identified as the same location.

Our approach goes in the same direction, in that it applies the idea of modeling location awareness with state machines *to any kind of context-related attribute*. We have proposed in [8] a stochastic extension of state machines, called *Context Evolution Models*[2] (CEM), where the context is modeled as a combination of three different *Context Sources* (CS), namely the physical location of users, the logical location of software components, and the status of hardware resources. In this paper, we generalize the modeling in [8] to any combination of different CSs. The assumption of our stochastic modeling framework is that each CS is represented by a *Context Attribute* (CA) that takes a finite set of values and whose evolution can be modeled through a CEM. Most importantly, we provide the formal underpinnings of our framework by defining a *composition semantics* for CEMs, which is inspired by stochastic process algebras as compositionality is a first-class notion in those formalisms.

As Fig. 1 shows, a context is a free combination (denoted by $\circ$) of CAs and their values (denoted by ν metavariables). In such a way, we are able to model the needed degrees of context awareness (denoted by *), from the simplest one, which considers only one CA (1-*awareness context* layer), to the whole context awareness described by combining all the identified CAs. Using stochastic state machines to model the evolution of each CA and leveraging on the state machine composition, we are able to represent the context evolution, whatever its degree is. In the following, we call *First-Order Context Evolution Model* (FOCEM) the stochastic state machine modeling a single CA associated with a known, unique CS, and *Higher-Order Context Evolution Model* (HOCEM) the state machine modeling the considered context obtained by combining the FOCEMs of the CAs forming the context itself.

In the rest of this section, we formalize the notion of FOCEM (Sect. 3.1) and the notion of HOCEM (Sect. 3.2). We discuss the basic concepts underlying them, their representation in terms of state machines, their composition mechanism inspired by stochastic process algebras, and their quantitative aspects based on continuous-time Markov chain theory, all of which will be exemplified on the MeH case study.

3.1 Context attributes and FOCEM

A FOCEM is a state machine, where each *state* represents part of the context and corresponds to a particular value in a finite set of homogeneously typed values that can be assigned to a specific CA. For example, percentages—whose value is

[2] We replace here the original name Manager with the more appropriate Context Evolution Model.

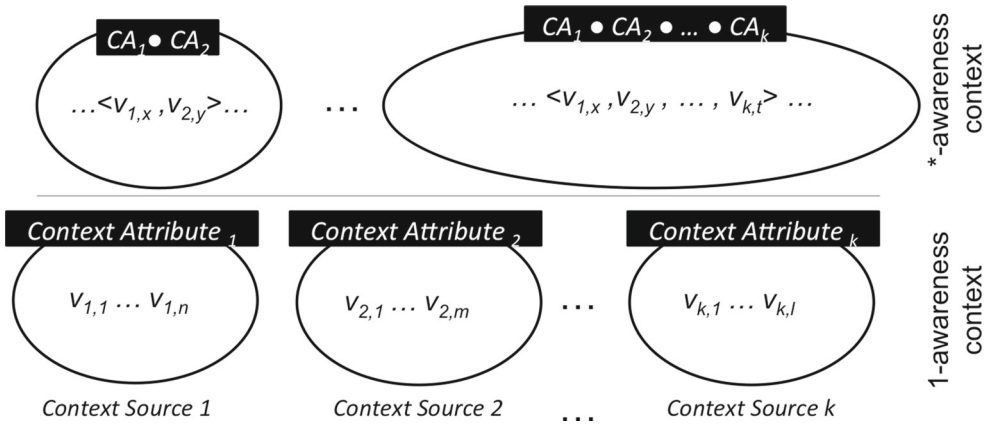


Fig. 1 Context as a combination of heterogeneous attributes

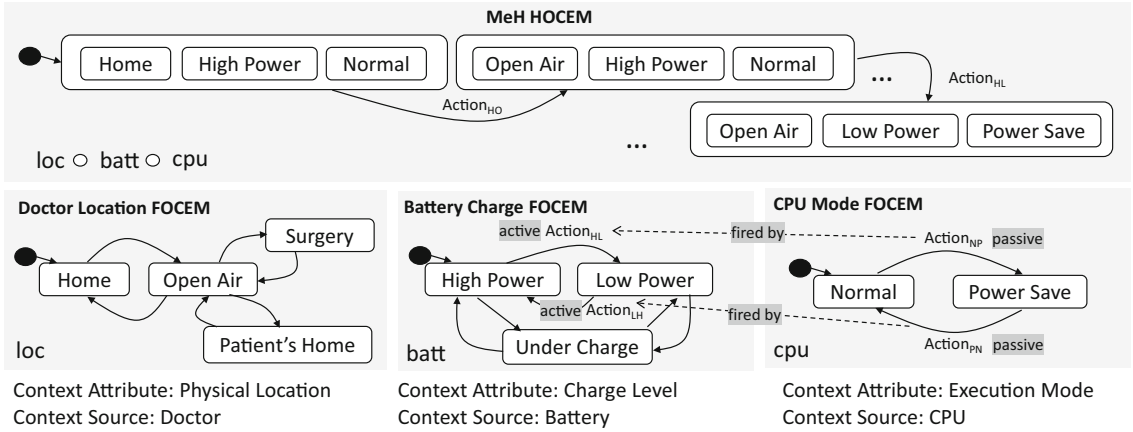


Fig. 2 MeH: HOCEM and FOCEMs for doctor, battery, and CPU elements

an integer number between 0 and 100—are normally used to describe the charge level of a battery.

A *transition* represents a change in such a typed value. It can be triggered by an internal action that modifies the current value of the CA, e.g., an increment/decrement of the percentage representing the battery charge level. A transition can also be triggered by a remote event occurring in a different FOCEM, e.g., a battery level change may cause a device screen to reduce its brightness.

Instances of FOCEMs for MeH are shown in the bottom layer of Fig. 2. In particular, this layer contains three FOCEMs related to three different CAs:

- The *Doctor Location FOCEM* represents the possible values (Home, Surgery, Open Air, and Patient's Home) of the physical location attribute for the doctor. Transitions here represent the possible moves of the doctor among locations [20].
- The *Battery Charge FOCEM* represents the charge level evolution of the battery of the doctor's PDA. A threshold (e.g., 25%) triggers the transition from *high-power* state to *low-power* one and vice versa.
- The *CPU Mode FOCEM* represents two execution modes for the CPU of the PDA used by the doctor, namely *normal* and *power save*. The latter is meant to reduce the power consumption by decreasing the CPU frequency.

Transitions are labeled with the corresponding actions and the related quantitative information. The action set is partitioned into *active* actions and *passive* actions. Although this action classification is inspired by analogous action classifications in stochastic process algebras such as PEPA [27] and EMPA [11] and architectural description languages based on them [5,12] to formalize master–slave synchronizations, here it is employed to support context dependencies in terms of cause–effect relationships from an active action of a FOCEM to a passive action of another FOCEM. For instance, the transition from *high power* to *low power* within the battery—which is labeled with an active action—induces the transition

from *normal* to *power save* within the CPU—which is labeled with a passive action.

Also, the quantitative aspects associated with actions are inspired by stochastic process algebras and hence rely on exponentially distributed durations coming from continuous-time Markov chain (CTMC) theory [45]. The reason for this choice is that it yields a simpler mathematical treatment supported by a number of well-established steady-state and transient-state analysis techniques. On the expressiveness side, it turns out that exponential distributions are adequate for modeling the timing of many real-life phenomena like arrival processes, failure events, and chemical reactions. Moreover, an exponential distribution is the most appropriate stochastic approximation in the case in which only the average duration of an activity is known [19]. Finally, proper combinations of exponential distributions, called phase-type distributions [42], can approximate most general distributions arbitrarily closely. As an example, a fixed duration d can be rendered by a sequence of n exponential phases each of rate λ such that $n/\lambda = d$; the greater the n, the better the approximation.

Every occurrence of an active action is governed by its execution rate $\lambda \in \mathbb{R}_{>0}$. This uniquely quantifies the duration of the action execution in terms of an exponentially distributed random variable, whose expected value $1/\lambda$ represents the average duration of the action. According to CTMC theory, whenever several active actions are enabled in a state, the action that is executed is the one that samples the least duration (race policy). As a consequence, every active action has an execution probability that is proportional to its rate. Moreover, the sojourn time in a state turns out to be exponentially distributed, with rate given by the sum of the rates of the active actions enabled in that state.

In contrast, every occurrence of a passive action is governed by its associated selection probability $p \in \mathbb{R}_{]0,1]}$. Whenever a state enables at least one occurrence of a certain passive action, it must be the case that the selection probabilities of all the occurrences of that passive action enabled in that state sum up to 1. If only one occurrence is enabled, then $p = 1$. This is the case with both passive actions in the CPU states of Fig. 2, as shown in the forthcoming Table 1 where also the rates of the various active actions are instantiated.

Definition 1 A FOCEM is a triple $\mathcal{FOCEM} = (S, A, \longrightarrow)$ where:

- S is a finite set of states.
- $A = A_a \cup A_p$ is a finite set of actions, where A_a is a set of exponentially timed active actions, while A_p is a set of probabilistic passive actions, such that $A_a \cap A_p = \emptyset$.
- $\longrightarrow \subseteq S \times ((A_a \times \mathbb{R}_{>0}) \cup (A_p \times \mathbb{R}_{]0,1]})) \times S$ is a transition relation between states, in which every transition is labeled with an action and a real number. ■

3.2 Context composition and HOCEM

A HOCEM is a model resulting from the composition of a set of FOCEMs representing the evolution of CAs of interest. It follows that:

- Whenever a FOCEM is added to the set, deleted from the set, or modified within the set, a new composition is computed to rebuild the HOCEM.
- The full context model is given by the HOCEM resulting from the composition of all the FOCEMs for the considered system.

Similar to CSs in Fig. 1, FOCEMs may need to be combined depending on the type of context awareness suitable for the considered application. A lumping process is then required that combines two or more FOCEMs to raise the context-awareness degree. Any combination of two or more FOCEMs generates a HOCEM in which every state represents an instance of the composite context of interest for the application, i.e., a set of heterogeneously typed values assigned to attributes from two or more CSs. As an example, through a HOCEM the MeH system may be aware of a doctor who is using the PDA to invoke the RPD service (1) while working at the surgery (2) with a low charged battery that (3) induces the CPU of the PDA to limit its frequency so to decrease the power consumption. The top layer of Fig. 2 represents an excerpt of a HOCEM for the MeH application.

Context dependencies are expressed across FOCEMs through a mechanism that we call *remote firing*, which is inspired by an analogous mechanism within Harel's statecharts [26]. A dependency that induces a remote firing is established by binding a *(firing) event* on a transition of a FOCEM to one or more *(fired) events* on transitions of one or more different FOCEMs, thus building typical cause–effect events. This mechanism can be used, for example, to model a change in the CPU frequency (the fired event) as induced by a change in the battery charge level (the firing event). Such a dependency is shown in Fig. 2 by two dashed *fired-by* arrows from the fired transitions of the CPU Mode FOCEM to the firing transitions of Battery Charge FOCEM. We call *static* a causal dependency explicitly declared by the modeler through an expression of the form $a \mapsto b$, where a is an (exponentially timed) active action of a FOCEM, while b is a (probabilistic) passive action of another FOCEM. Although only cause–effect pairs can be directly modeled, cause–effect chains emerge due to the fact that FOCEMs also describe the continuation after the execution of the actions involved in the cause–effect pairs.

The formal definition of the composition of FOCEMs based on remote firing relies on the classification of their actions into (exponentially timed) active actions and (probabilistic) passive actions. In the field of stochastic pro-

cess algebras, from which it is taken, this classification is used to enforce multiaction synchronizations according to the generative–reactive cooperation mechanism [11]. This means that, among the identically named actions participating in a synchronization, one of them must be active while all the others must be passive. The overall rate of the synchronization is given by the rate of the active action multiplied by the product of the selection probabilities of the involved passive actions, which is consistent with the fact that it is the active action that triggers the passive ones.

However, the synchronization mechanism of stochastic process algebras is too rigid for our purposes, because it is based on fixed synchronization sets. For example, if we have three processes with synchronization set $\{a\}$ between any pair of them, then all of them must synchronize on a—which is possible only if a is enabled in all of them—thereby excluding the possibility that only two of them synchronize on a at a certain point in time. For an adequate modeling of context evolution, we need a more flexible mechanism in which the execution of an active action a of a FOCEM with static causal dependencies $\{a \mapsto b_i \mid i \in I\}$ is not blocked by the fact that for some $j \in I$ passive action b_j is not enabled by FOCEM j.

We call *dynamic* a causal dependency arising from the statically declared ones, and we express it as $a \mapsto B$ where a is an active action of a FOCEM, B is a set of passive actions each belonging to a distinct FOCEM, and for each $b \in B$ the static causal dependency $a \mapsto b$ has been declared. Given a dynamic dependency $a \mapsto B$, when the active action a is enabled, the execution of a induces the simultaneous execution of the *maximal* subset B' of passive actions in B, such that all those actions are enabled at that point in time in their respective FOCEMs. Notice that action a is executed even if $B' = \emptyset$. The execution of $a \mapsto B'$ causes the local states of the FOCEMs enabling actions in $\{a\} \cup B'$ to advance, while all the other FOCEMs stay idle.

Definition 2 Given $n \in \mathbb{N}_{\geq 2}$, let:

- $\mathcal{FOCEM}_i = (S_i, A_i, \longrightarrow_i)$ be a FOCEM for all $i = 1, \ldots, n$, with $A_i \cap A_j = \emptyset$ for $i \neq j$.
- SD_i be a possibly empty set of static causal dependencies declared for $\mathcal{FOCEM}_i$, each being expressed as $a \mapsto b$ where $a \in A_{i,\mathrm{a}}$, $b \in A_{j,\mathrm{p}}$, and $i \neq j$.

The corresponding HOCEM is a quadruple $\mathcal{HOCEM} = (S, A, DD, \longrightarrow)$ where:

- $S = S_1 \times \cdots \times S_n$ is the set of composite states.
- $A = A_\mathrm{a} \cup A_\mathrm{p}$ is the set of actions, where $A_\mathrm{a} = \bigcup_{1 \leq i \leq n} A_{i,\mathrm{a}}$ while $A_\mathrm{p} = \bigcup_{1 \leq i \leq n} A_{i,\mathrm{p}}$.
- DD is the set of dynamic causal dependencies of the form $a \mapsto B$ such that:
  - $a \in A_\mathrm{a}$.
  - $B \subseteq A_\mathrm{p}$.
  - For all $b \in B$, $a \mapsto b \in \bigcup_{1 \leq i \leq n} SD_i$.
  - For all $b_1, b_2 \in B$ such that $b_1 \neq b_2$, if $b_1 \in A_{i,\mathrm{p}}$ then $b_2 \in A_{j,\mathrm{p}}$ with $j \neq i$.
- $\longrightarrow \subseteq S \times (DD \times \mathbb{R}_{>0}) \times S$ is a transition relation between composite states containing transitions of the form $s \xrightarrow{a \mapsto B', \lambda} s'$, where $s = (s_1, \ldots, s_n)$ and $s' = (s'_1, \ldots, s'_n)$, such that:
  - There exist $i_a \in \{1, \ldots, n\}$ and $\lambda_a \in \mathbb{R}_{>0}$ such that $s_{i_a} \xrightarrow{a, \lambda_a}_{i_a} s'_{i_a}$.
  - B' is a maximal subset of A_p such that, for all $b \in B'$, there exist $j_b \in \{1, \ldots, n\}$ and $p_b \in \mathbb{R}_{]0,1]}$ such that $s_{j_b} \xrightarrow{b, p_b}_{j_b} s'_{j_b}$.
  - $\lambda = \lambda_a \cdot \prod_{b \in B'} p_b$, with the second factor being 1 when $B' = \emptyset$.
  - For all $i \in \{1, \ldots, n\}$ such that $i \neq i_a$ and $i \neq j_b$ whenever $b \in B'$, $s'_i = s_i$. ■

From a HOCEM, we can easily obtain a quantitative model in the form of a continuous-time Markov chain (CTMC) [45]. This is essentially derived by first eliminating dynamic causal dependencies from transition labels and then merging all the transitions between any pair of composite states into a single transition, whose rate is the sum of the rates of the original transitions as a consequence of the race policy. By solving this CTMC, we can derive the distribution of the probabilities of being in the various states of a HOCEM at a certain time.

More precisely, the aforementioned CTMC can be represented as a state-indexed matrix $\boldsymbol{Q}$ called the infinitesimal generator. The entry $q_{h,k} \in \mathbb{R}_{\geq 0}$, $h \neq k$, represents the rate at which it is possible to go from composite state s_h to composite state s_k through a single transition, while $q_{h,h}$ is set to $-\sum_{k \neq h} q_{h,k}$ thus causing all rows to sum up to 0. The solution of the CTMC in matrix form is computed as follows:

- Given the initial probability distribution $\boldsymbol{\pi}(0)$ over composite states, the transient solution $\boldsymbol{\pi}(t)$ at time $t \in \mathbb{R}_{>0}$ is obtained by solving the differential equation system:

$$\boldsymbol{\pi}(t) \cdot \boldsymbol{Q} = \frac{\mathrm{d}\boldsymbol{\pi}(t)}{\mathrm{d}t}$$

  which can be done by means of standard techniques like uniformization.
- The stationary solution $\boldsymbol{\pi} = \lim_{t \to \infty} \boldsymbol{\pi}(t)$ is obtained (if any) by solving the linear equation system:

$$\boldsymbol{\pi} \cdot \boldsymbol{Q} = \mathbf{0}, \quad \sum_{s \in S} \pi[s] = 1$$

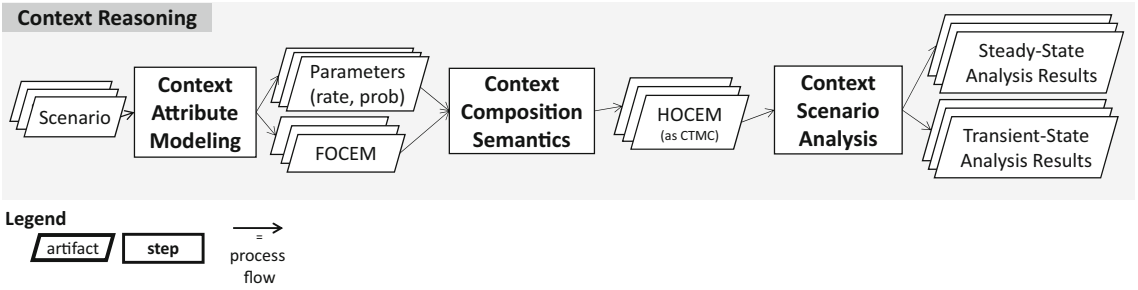


Fig. 3 Context reasoning workflow

which can be done by means of standard techniques from linear algebra.

4 Reasoning on the context

In this section, we show the reasoning capabilities on the context and its evolution that are enabled by our framework.

For a schematic view of our approach, Fig. 3 depicts the context reasoning workflow, whose steps will be described in the following subsections.

The CA modeling step for two possible scenarios, which we denote by ScA and ScB, is described in Sect. 4.1. Then, Sect. 4.2 shows: (1) the application of context composition semantics on FOCEMs and their parameters, as obtained in the previous step, to generate a CTMC-based HOCEM for each scenario, and (2) the results of the steady-state and transient-state context scenario analyses on these HOCEMs. Finally, Sect. 4.3 carries out a sensitivity analysis of the ScA scenario.

4.1 Context attribute modeling

We are able to represent, through CEMs, different context scenarios for a context-aware application. Each scenario comprises a set of basic FOCEMs, one for each attribute that the particular application is aware of.

For sake of illustration, we consider again the MeH case study and we combine three FOCEMs (Doctor Location FOCEM, Battery Charge FOCEM, and CPU Mode FOCEM) that appear at the bottom of Fig. 2.

We assume that in both scenarios, ScA and ScB, the MeH application is capable of sensing the same CAs, which are: (1) the physical location of the doctor, (2) the charge level of the battery equipping the doctor's PDA, and (3) the execution modes of the CPU on the same PDA.

Refined versions of the FOCEMs in the bottom layer of Fig. 2 are depicted in Fig. 4 using a UML-like state machine diagram notation and are detailed below:

- *Doctor Location FOCEM*. Figure 4a shows the states and transitions of the FOCEM that models the evolution of the physical location CA. The doctor stays at *home* while not working or at the *surgery* and at the *patients' homes* while giving assistance. The doctor can move among such places in any direction, and thus, an *open air* state is placed between all pairs of locations.
- *Battery Charge FOCEM*. The battery charge level of the doctor's PDA may assume values that vary from a minimum of 0 to a maximum of 100. In order to limit the number of states, we set up a threshold of 25% to distinguish the *low-power* state (0–24) from the *high-power* state (25–100). In addition, a battery reaches an *under charge* state when it is plugged into a power socket. Figure 4b shows the states and transitions of the FOCEM that models the evolution of the charge level CA. In our scenarios, we assume that the doctor can recharge the PDA battery only at home and at the surgery. When leaving these places, the PDA has to be unplugged if it is *under charge*. Therefore, directed dotted lines are drawn from the outgoing (fired) transitions of the *under charge* state of the Battery Charge FOCEM and the (firing) outgoing transitions from the *home* and *surgery* states of the Doctor Location FOCEM.
- *CPU mode FOCEM*. The CPU of the PDA may work in two execution modes: *normal*, i.e., without any restriction on the clock frequency, and *power save*, when the system needs to reduce the power consumption because the battery has reached the low-power state. Figure 4c shows the states and transitions of the FOCEM that models the evolution of the CPU mode CA. Two directed dotted lines are drawn from the fired transitions of the CPU Mode FOCEM to the corresponding firing transitions of the Battery Charge FOCEM to represent the remote firings among them.

Transitions in Fig. 4 are labeled with action names, conditions (where needed), and $-prefixed variables for rates and probabilities. We remind that the use of rates of exponential distributions for characterizing durations is justified by a number of reasons that are already expressed in Sect. 3.1.

The two considered scenarios, ScA and ScB, differ for the stochastic parametrization of these three FOCEMs. There-

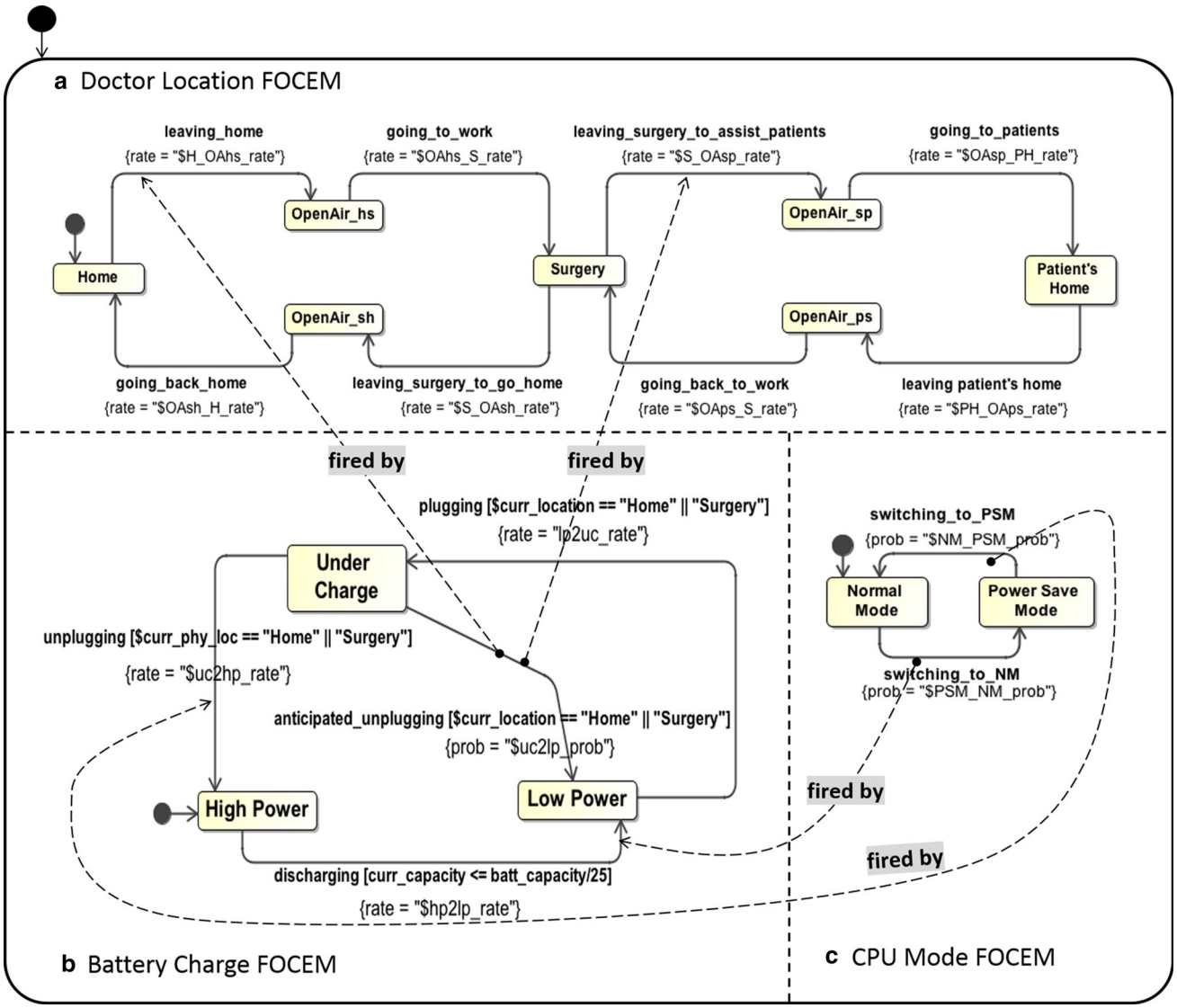


Fig. 4 A combined view of the FOCEMs for **a** doctor's physical location, **b** charge level of PDA battery, and **c** execution modes for PDA CPU

fore, the specification of the two scenarios consists of distinct rates/probabilities for active/passive actions. Table 1 reports such input parameters of ScA and ScB.

The rows in Table 1 are grouped in three sets, one for each FOCEM. Each row includes an *action name*, an *action type* (active or passive), a *parameter* name and *type* (rate or probability), and the values for such parameters in ScA and ScB.

Rates for Doctor Location FOCEM in ScA come from assuming that the doctor[3]:

- Spends 10 h at home before leaving.
- Takes half an hour to move from home to the surgery.
- Remains at work 8 h before going home.
- Leaves the surgery every 4 h by ambulance to reach patients' homes.
- Goes to patient's home by ambulance in 20 min.
- Gives first aid to the patient in 5 min.
- Goes back to the surgery by ambulance in 20 min.
- Goes back home at the end of the working day in 1 h, assuming that the doctor stops somewhere before arriving at home to perform an activity (e.g., going to the gym or shopping).

In ScB, the rates of (active) actions have been modified to represent a doctor who spends less time at home and at the surgery and moves more frequently, thus spending more time on the paths in the open air connecting such places. The assumptions that differentiate ScB from ScA consider that the doctor:

[3] All numbers mentioned in the following represent mean values used to obtain rates of actions.

Table 1 Input parameters for the FOCEMs in ScA and ScB

Action name	Type	Parameter	Type	Value of rate or probability			
				ScA		ScB	
				Times/h	Prob.	Times/h	Prob.
Doctor location FOCEM							
Leaving_home	Active	$ H_OAhs_rate	Rate	0.10		0.13	
Going_to_work	Active	$ OAhs_S_rate	Rate	2.00		1.00	
Leaving_surgery_to_assist_patient	Active	$ S_OAsp_rate	Rate	0.25		0.50	
Going_to_patients	Active	$ OAsp_P_rate	Rate	3.00		3.00	
Leaving_patient's_home	Active	$ P_OAps_rate	Rate	12.00		12.00	
Going_back_to_work	Active	$ OAps_S_rate	Rate	3.00		3.00	
Leaving_surgery_to_go_home	Active	$ S_OAsh_rate	Rate	0.13		0.25	
Going_back_home	Active	$ OAsh_H_rate	Rate	1.00		0.50	
Battery charge FOCEM							
Discharging	Active	$ HP_LP_rate	Rate	0.18		0.22	
Plugging	Active	$ LP_UC_rate	Rate	0.53		0.66	
Unplugging	Active	$ UC_HP_rate	Rate	0.50		0.50	
Anticipated_unplugging	Passive	$ UC_LP_prob	Prob		1		1
CPU mode FOCEM							
Switching_to_PSM	Passive	$ NM_PSM_prob	Prob		1		1
Switching_to_NM	Passive	$ PSM_NM_prob	Prob		1		1

– Spends 8 h at home before leaving.
– Takes 1 h (twice than in ScA) to move from home to the surgery.
– Remains at work 4 h (half than in ScA) before going home.
– Leaves the surgery every 2 h (again, two times per day).
– Goes back home at the end of the working day in 2 h (twice than in ScA).

For the Battery Charge FOCEM, we imagine two different usages of the PDA in ScA and ScB, where the former induces a lower power consumption than the latter.

In both scenarios, we assume that the RPD service will be always running on the doctor's PDA, i.e., we do not consider the stand-by time. We also assume a linear power consumption, while the RPD service is running. In ScA, a fully charged battery is consumed in 7 h and a half, while in ScB, the autonomy of the PDA is reduced to 6 h and 20 min. Accordingly, we calculate the time needed to run down the battery: (1) until a certain threshold that we set to 25% of the total capacity of the battery and (2) from such a threshold until the battery lasts.

The parametrization of the CPU Mode FOCEM does not change across scenarios because the actions labeling the transitions from normal mode to power save mode and vice versa are passive. Since each of them is the unique outgoing transition from the corresponding state, a probability equal to 1 is assigned in all cases.

The values of all the above-defined parameters have been estimated by looking at multiple real-life examples of doctor behavior profiles from different sources. This observation brought us to synthesize the mean values (i.e., times or rates) used in this paper. These values are subject to inaccuracies, mostly due to the variance among behaviors of profiled doctors. The combination of such inaccuracies, of course, can propagate to the analysis results. However, our approach is intended to be used at system design phase, when the analysis is aimed at taking design decisions through comparisons of different alternatives. In this direction, the inaccuracies of results can be mitigated by the fact that multiple sets of values can be assigned to parameters in order to compare analysis results and figure out relevant trends, as we will do in Sect. 4.3. Decisions that induce unsatisfactory trends can be highlighted as critical ones in the design process. Therefore, our approach is not intended to take fine-grain decisions, it is rather aimed at comparing different situations and, on the basis of numerical results, at providing support for context-aware system design decisions before a system is implemented, with particular emphasis on the early phases of its life cycle.

4.2 Context composition semantics and scenario analysis

The given set of FOCEMs can be composed of different HOCEMs. From a modeling perspective, a HOCEM is a com-

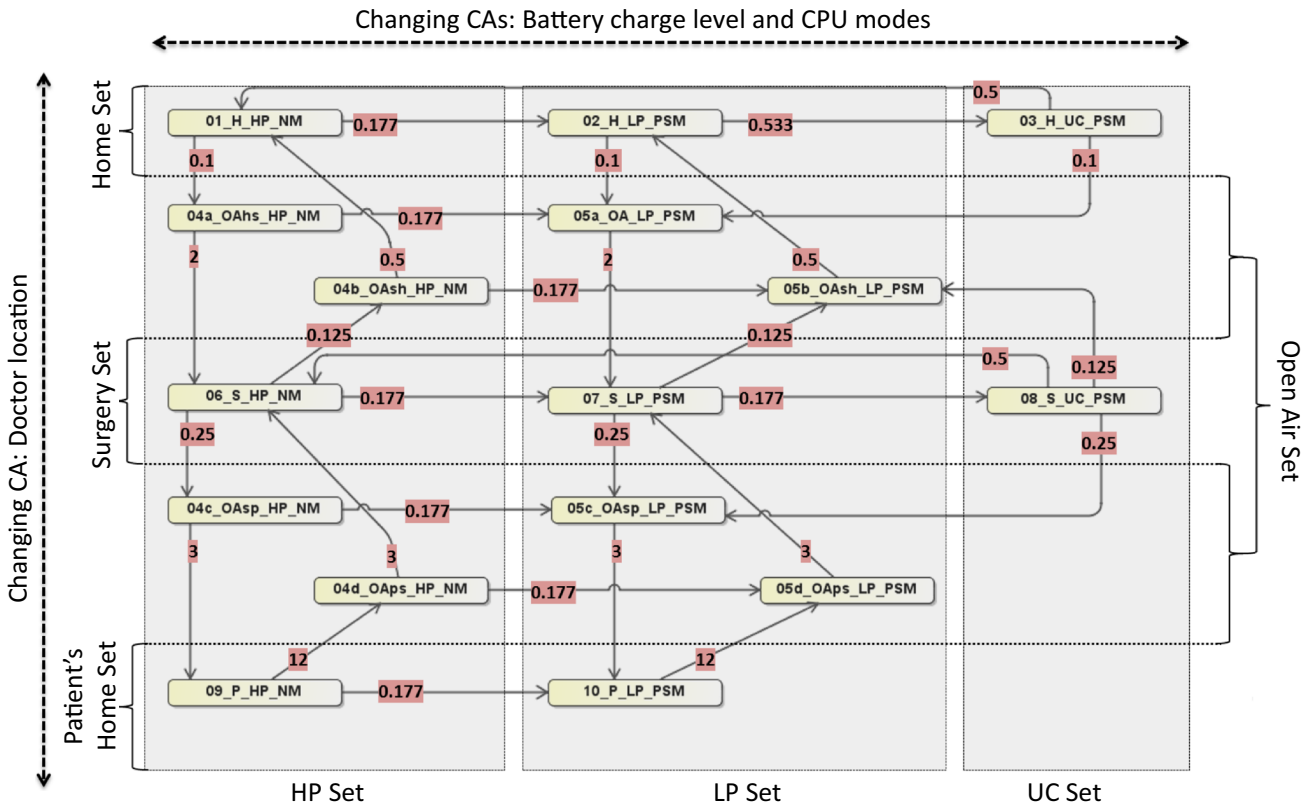


Fig. 5 The HOCEM for MeH parametrized with rates for ScA

posite state machine that executes its constituting FOCEMs in parallel. In Fig. 4, the constituting FOCEMs were represented within distinct regions delimited by dashed lines.

In Fig. 5, we show the HOCEM state machine obtained from the context composition semantics (see Sect. 3.2) applied to the FOCEMs in Fig. 4. This HOCEM explicitly represents composite states and their transitions, as well as the action rate annotations, which in Fig. 5 refer to the ScA scenario.[4]

In order to illustrate the potential reasoning capabilities of our approach, the resulting composite context states and transitions of the HOCEM have been arranged to facilitate both left-to-right (or right-to-left) and top-down (or bottom-up) readings, as indicated by the dotted arrows external to the figure.

The horizontal arrow follows the coupled evolution of two CAs, which are the battery charge level (high power, low power, and under charge) and the consequent execution modes of the CPU (normal mode, power save mode), both equipping the doctor's PDA.

The vertical arrow follows the context evolution with respect to the physical moves of the doctor across physical places (i.e., home, surgery, patients' home, and the paths in the open air).

In Fig. 5, the context states have been horizontally and vertically partitioned in different sets. States belonging to the same set share a common CA value. In this respect, the vertical sets in Fig. 5 focus on the evolution of the physical locations of the doctor while binding the other two context attributes (i.e., battery charge level and CPU modes):

- HP Set identifies a high-power context set where the RPD service is always running on a fully charged PDA, and the only CA that varies is the physical location of the doctor equipped with the PDA.
- LP Set identifies a set of low-power contexts where the RPD service is always running on a PDA where the charge level of the battery is equal to or lower than the chosen threshold, namely in our example 25%.
- UC Set includes the two context states where the battery is under charge. According to the remote firing dependencies among the transitions of the Doctor Location and Battery Charge FOCEMs, a recharge operation happens only at doctor's home or at the surgery. When the doctor leaves such two places to go home or to patients' homes, the recharge is interrupted (i.e., the PDA is unplugged from the socket plug), and according to the charge level

[4] For sake of readability, we do not show any action name but only the resulting rate.

of the battery, a new context state belonging to the HP Set or LP Set is reached.

Similarly, the horizontal sets focus on the combined evolution of two CAs, that are the charge level of the battery and the execution modes of the CPU, while binding doctor's physical location:

- Home Set includes the context states where the doctor stays at home.
- Surgery Set includes the context states where the doctor works at the surgery.
- Patient's Home Set includes the context states in which the doctor is giving assistance to the patients at their homes.
- Open Air Set includes the context states where the doctor moves outdoors between the aforementioned places.

Since the states of a HOCEM are obtained from the Cartesian product of the corresponding FOCEMs states, the size of the HOCEM state space quickly grows with the number of considered CAs and the number of states of each attribute. Therefore, the analysis of a HOCEM can become computationally infeasible due to scalability issues in case of systems with many and complex CAs. This problem is mitigated by the fact that, even in systems with numerous attributes, the analysis of a complete HOCEM (i.e., the one that results from the combination of all FOCEMs) is often meaningless, because it is difficult to identify causes of problems when too many attributes are considered at the same time. Indeed, our approach is modular, as it does not mandate to consider all attributes together, but selective combinations of attributes can be considered to synthesize a partial HOCEM. As evidenced in this section, the aggregation of states allows us to highlight the role that different attributes may have in partial context analyses that focus only on some CAs while leaving hidden the other ones. Without grouping states in sets as in Fig. 5, all the context states and their attributes are instead equally important throughout the context analysis.

In Appendix A, we show the tables containing the transition rates of the HOCEMs for both ScA and ScB scenarios. By analyzing the CTMCs corresponding to the parametrized HOCEMs for ScA and ScB, we obtain the steady-state and transient-state probabilities for the context states shown in Fig. 4. These numbers are at the basis of the computation of performability measures. It is worth recalling that within our framework there are also other analysis techniques that can be applied, most notably probabilistic/stochastic model checking [4], but they will not be described as they are outside the scope of this paper.

In the remainder of this section, we analyze the state probabilities of the HOCEM obtained for the MeH case study, under the two previously defined scenarios. We have identified scenarios that describe two quite different situations in terms of doctor mobility, so to demonstrate that our approach supports the non-functional analysis of different real-life situations. Many other scenarios may need to be analyzed in order to compare the effects of system design decisions under different contexts, and this can incur in scalability problems. However, since the scenarios differ in terms of transition probabilities, this problem can be mitigated by analyzing first extreme cases and then by generating intermediate scenarios only where critical situations have been identified. For example, critical situations may be originated by conflicting results on the same HOCEM under different scenarios. In these cases, designers and domain experts should interact to take (sometime heavy) decisions, such as preventing some context states from being reached under certain circumstances.

4.2.1 Steady-state analysis

In the rows of Table 2, we report all 16 context states of Fig. 5. For each of them, steady-state probabilities for ScA and ScB have been obtained by solving the CTMCs corresponding to the HOCEMs, and the rightmost column shows the differences between the two scenarios.

We have emphasized the three most visited context states in each scenario, because they could allow to identify whether most of the time, at the steady state, is spent in contexts with some peculiarities. Table 2 shows this evidence for ScB, where the three highly visited states are the ones where the doctor stays at home. This information can be exploited for multiple purposes, such as improving application aspects that can be appreciated in a home environment in the ScB case. For example, high-definition images can be shown due to the likely availability of a HD screen. The same evidence is not provided for ScA, where the highly visited states do not have much in common, apart from not representing outdoor contexts. However, both scenarios share the most visited state, which is the one in the first row of the table, meaning that any improvement of the application in such a context would be beneficial for both context scenarios.

We have also emphasized, in the rightmost column, the three highest variations of sojourn probabilities in ScB with respect to ScA. This observation goes in the opposite direction with respect to the latest comment on steady-state probabilities, because the three identified rows correspond to states that are peculiar only for one scenario. Hence, it can be appropriate to act on the application characteristics in these contexts only if accurate data are available about the scenario occurrence. For example, the highest difference is obtained for context state 06. If it can be asserted that the application will be used in ScA, then it is worth to tailor the application to such a context. In the opposite case, namely if no clue is given about the occurrence of either ScA or ScB,

Table 2 Steady-state probabilities for ScA and ScB

Context state				Steady-state probability vector				Diff.	
No.	Phy.	Batt.	CPU	ScA		ScB			
01	H	HP	NM	1st	0.2567	1st	0.2334		−0.0233
02	H	LP	PSM		0.1162	2nd	0.1223		0.0061
03	H	UC	PSM		0.1033	3rd	0.1292	3rd	0.0259
4a	OAhs	HP	NM		0.0118		0.0239		0.0121
4b	OAsh	HP	NM		0.0195		0.0319		0.0124
4c	OAsp	HP	NM		0.0144		0.0143		−0.0001
4d	OAps	HP	NM		0.0134		0.0130		−0.0004
5a	OAhs	LP	PSM		0.0120		0.0367		0.0247
5b	OAsh	LP	PSM		0.0281		0.0893	2nd	0.0612
5c	OAsp	LP	PSM		0.0173		0.0261		0.0088
5d	OAps	LP	PSM		0.0183		0.0274		0.0091
06	S	HP	NM	2nd	0.1835		0.0918	**3rd**	−0.0917
07	S	LP	PSM	3rd	0.1227		0.0986		−0.0241
08	S	UC	PSM		0.0748		0.0520		−0.0228
09	P	HP	NM		0.0036		0.0035		−0.0001
10	P	LP	PSM		0.0044		0.0066		0.0022
Total:					1.0000		1.0000		0.0000

Table 3 Steady-state analysis for subsets of context states for ScA and ScB

Analysis sets	Steady-state probability vector		Diff.
	ScA	ScB	
Home set	0.4762	0.4848	0.0087
Open air set	0.1349	0.2626	0.1277
Surgery set	0.3810	0.2424	−0.1385
Patient's home set	0.0079	0.0101	0.0022
Total	1.0000	1.0000	0.0000
HP set	0.5029	0.4118	−0.0911
LP set	0.3191	0.4070	0.0879
UC set	0.1780	0.1812	0.0032
Total:	1.0000	1.0000	0.0000

application improvements in this context could represent a useless effort for application designers.

To better understand the differences between the two scenarios, we further split the flattened HOCEM in Fig. 5 in different sets. We then calculate the steady-state probabilities on the different subsets of the HOCEM. The results for both scenarios are reported in Table 3, where we first observe that the MeH system provides its services mostly when the doctor is at home (around 0.48 both in ScA and ScB). However, in ScB the sojourn probability with respect to ScA: (1) in the open air doubles up to 0.26, (2) in the surgery decreases down to 0.24, and (3) it does not significantly vary at patient's home.

In ScB, we also assume a higher battery consumption (see Table 1) that, combined with the higher mobility of the doctor, leads to increase the sojourn probability in the LP Set. This means that the system will more likely cope with a resource-constrained context where the charge level of the battery is low and the computational power of the CPU is consequently reduced.

4.2.2 Transient-state analysis

For the same scenarios ScA and ScB, we carried out a transient analysis to obtain the sojourn probabilities in the context states of Fig. 5 at a certain instant t, and they are reported on the charts in Figs. 6 and 7. On the x-axis, the transient-state probabilities have been calculated 1, 6, 12, and 24 h before the CTMC reaches the steady state. In both scenarios, the starting context state of our transient analysis is the doctor at home with a fully charged battery (first row in Table 2). Sojourn probabilities of some context states are reported on the y-axis. We grouped the sojourn probabilities sets focusing on the physical location of the doctor (i.e., Home Set, Open Air Set, Surgery Set, and Patient's Home Set). Such sets are identified along the y-axis on the left and right sides of both charts to highlight their relative contribution to the sojourn probability, while t flows from left to right. Below each set name, we reported their transient-state probabilities at $t = 1$ (on the left) and at steady state (on the right). It is worth noting that, as expected, the steady-state probabil-

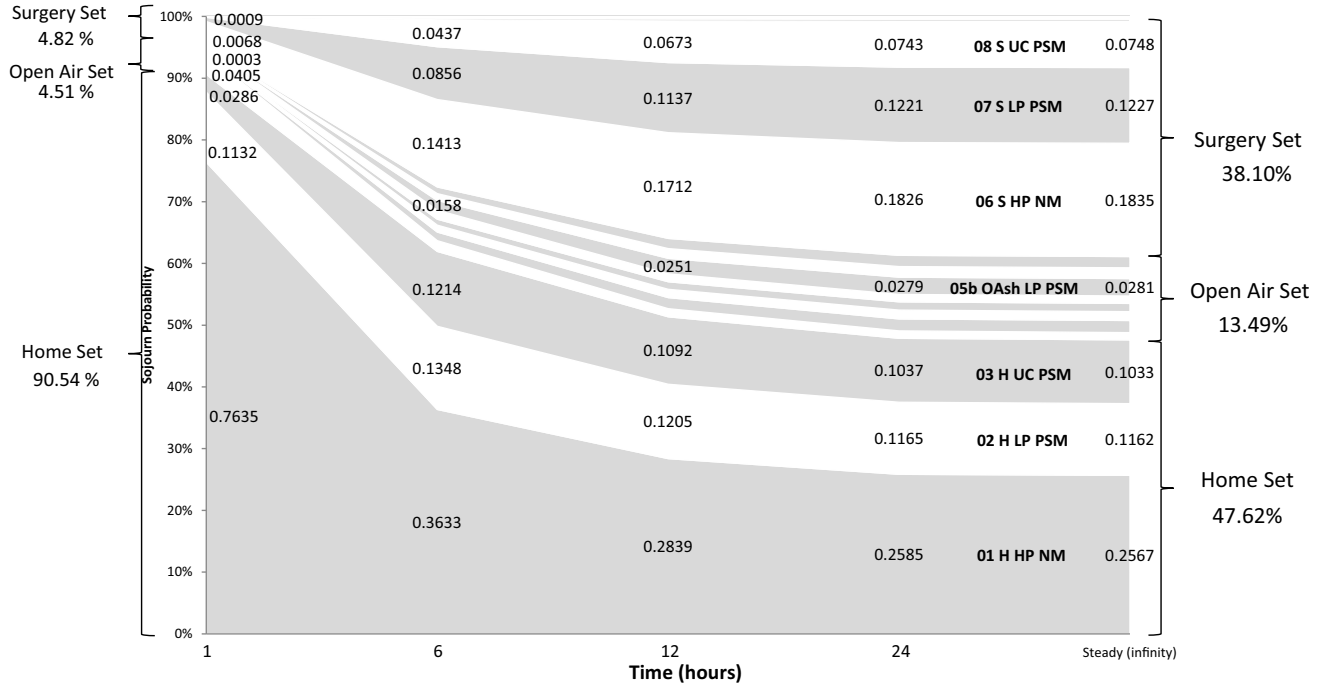


Fig. 6 Transient-state probabilities for ScA

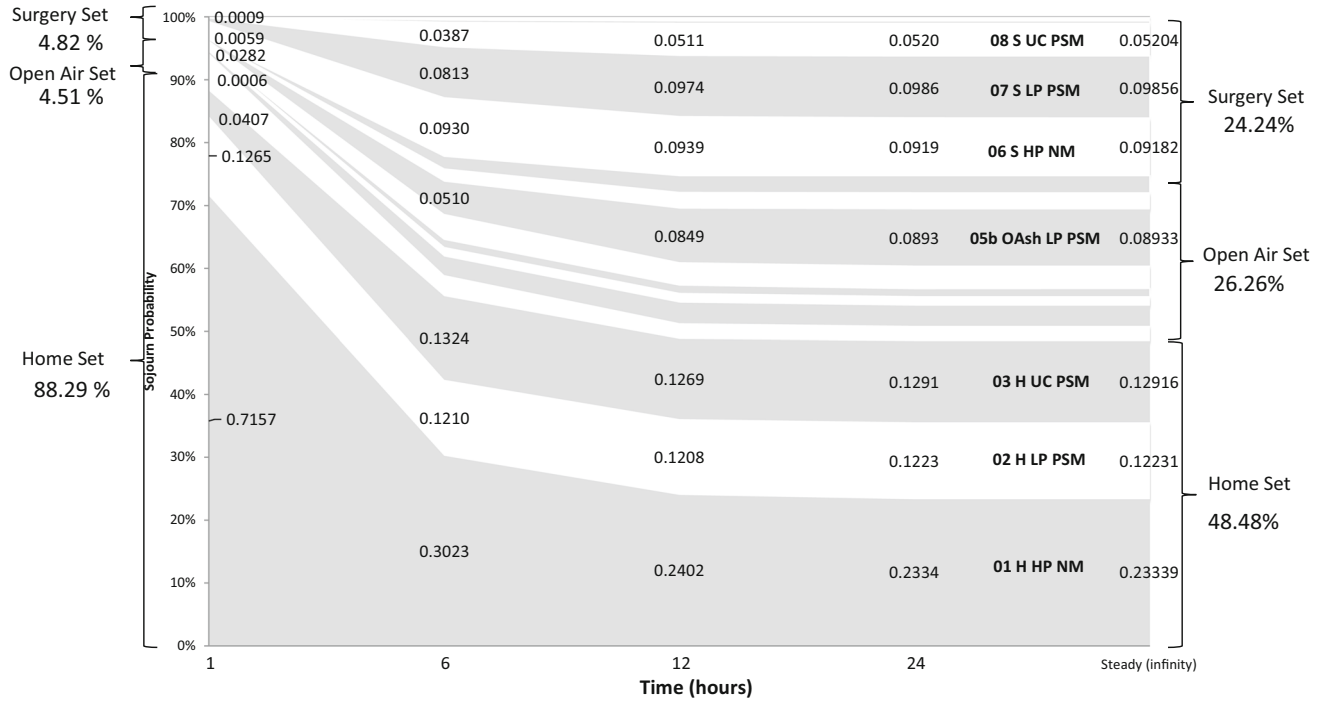


Fig. 7 Transient-state probabilities for ScB

ities are reached around $t = 24$ h in both scenarios, as this represents a periodical behavior of the doctor.

For ScA in Fig. 6, the Home and Surgery Sets are predominant. In both sets, the doctor experiences the best possible resource environment (i.e., high power and normal mode for battery and CPU, respectively) for most of the time. The contribution of the Open Air Set is always lower than 13.5%, whereas the one of the Patient's Home Set is always negligible, so it is not represented in Fig. 6.

The transient-state probabilities for ScB are depicted in the chart in Fig. 7. The evolution of the sojourn probabilities is similar to ScA, and the predominant states are still the ones in Surgery and Home Sets. However, the contribution of the Open Air Set increases at any time t due to the higher mobility assumed in ScB, going from 13.49% in ScA to 26.26% in ScB at $t = 24$. Within the Open Air Set, the highest increment in sojourn probabilities appears on context states with a low battery charge level (like context state 05*b*) since, by assumption, the doctor cannot charge the PDA outside home and surgery locations.

The relative relevance of the Home Set decreases, while the sojourn times in context states belonging to the Open Air Set continuously increase from 13.49% in ScA to 60.60% in ScB. Indeed, in ScB it may happen that a doctor has to leave the surgery to assist patients at their homes during the first hour of MeH usage, as specified by *leaving_surgery_to_assist_patient* row in Table 1, with a 0.5 times/h rate. In contrast, the same action is not likely to happen in ScA during the first hour of usage, due to a 0.25 rate. However, both scenarios assume that if the MeH system is started at home, it is very likely to happen that, after 1 h of usage, the doctor is still at home (i.e., *leaving_home* row in Table 1, with a 0.1–0.13 times/h rate). Therefore, it is very likely that the charge level of the battery is higher than the assumed threshold, i.e., 25%, during the first hour of usage; thus, this resource will likely remain in the high-power state. Finally, it is worth noting that the decrease in the relative relevance of the high-power contexts during the first 6 h of usage in ScB (i.e., $t = 6$ in Fig. 7) is also influenced by: (1) the 1 h faster discharging time of the battery (see *discharging* row in Table 1, with a 0.18 times/hour rate in ScA and 0.22 rate in ScB), and (2) the lower availability of plugs for starting a recharge (by assumption, *plugging* action is only available at home and at the surgery).

4.3 Sensitivity analysis

In order to show how the context analysis results may vary upon variations of context parameters, in this section we report a sensitivity analysis of the ScA scenario. The goal is to evaluate the impact of duration of doctor's activities, like assisting patients at home, and the impact of PDA battery capacity. From a modeling perspective, it corresponds to changing rates assigned to (active) actions *leaving_patient's_home* and *discharging* of the Doctor Location and Battery Charge FOCEMs, respectively, whose original values appear in Table 1.

In ScA, we assume that the doctor spends 5 min on average at patients' homes before leaving to move back to the surgery, i.e., before triggering the action *leaving_patient's_home* outgoing the patient's home state (see Fig. 4). For sake of sensitivity analysis, we extend here the visit duration up to 20 min by steps of 5 min, thus obtaining four alternatives. These variations are obtained by setting the corresponding λ parameter of the *leaving_patient's_home* exponential distribution to 12 (i.e., HOCEM ScA, our baseline for comparisons), 6, 4, and 3, respectively.

In ScA, the average time required to discharge a fully charged PDA battery (100%) down to the given threshold (25%) is 5 h and 30 min ($\lambda = 0.18$). Again for sake of sensitivity analysis, we increase up to 9 h and 30 min the time required to discharge the battery down to the threshold, by steps of 2 h. We obtain three alternative durations determined by the three values for the λ parameter 0.18 (i.e., HOCEM ScA, our baseline for comparisons), 0.13, and 0.1 assigned to the *discharging* action, respectively.

The combination of these alternatives generates a total of twelve variants of the ScA scenario, including the baseline case, which are listed in Table 4. For sake of sensitivity analysis, we have executed twelve times the context reasoning workflow illustrated in Fig. 3, once for each variant. From

Table 4 Configurations of variants

	Rate for actions: *leaving_patient's_home discharging*	Duration (h:mm)
Baseline		
HOCEM ScA	12	00:05
	0.18	05:30
Variants		
HOCEM 1	6	00:10
	0.18	05:30
HOCEM 2	4	00:15
	0.18	05:30
HOCEM 3	3	00:20
	0.18	05:30
HOCEM 4	12	00:05
	0.13	07:30
HOCEM 5	6	00:10
	0.13	07:30
HOCEM 6	4	00:15
	0.13	07:30
HOCEM 7	3	00:20
	0.13	07:30
HOCEM 8	12	00:05
	0.1	09:30
HOCEM 9	6	00:10
	0.1	09:30
HOCEM 10	4	00:15
	0.1	09:30
HOCEM 11	3	00:20
	0.1	09:30

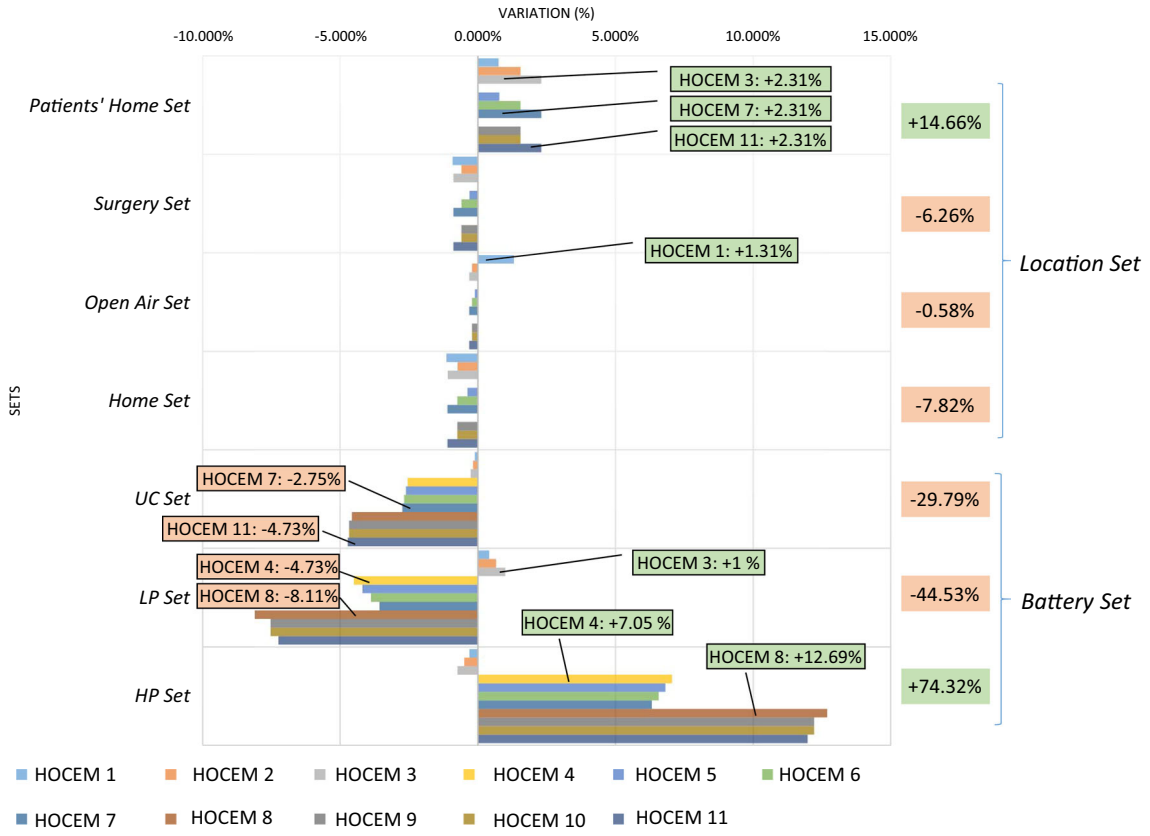


Fig. 8 Sensitivity analysis: sojourn probability variations in ScA

a modeling perspective, we have obtained twelve triples of FOCEMs and related parameters, and we have generated twelve corresponding HOCEMs. Since we do not change state, transition, and dependency sets, these HOCEMs are structurally identical to the one in Fig. 5, but they have different rates that come from the semantic composition of the FOCEMs with different parameters.

In Fig. 8, we report the analysis results. We reuse the logical partition of HOCEM in sets, as depicted in Fig. 5, and plot the variations in sojourn probabilities with respect to the HOCEM ScA scenario (1) for each set as a whole (on the right side) and (2) for each HOCEM variant, from HOCEM 1 to HOCEM 11. For sake of readability, the bar chart highlights, for each set, only the HOCEM x variants with the highest positive and negative variation (in percentage) with respect to the results of the ScA scenario (i.e., the first row in Table 4).

As expected, the sojourn probabilities in HOCEM states belonging to the HP Set and Patient's Home Set increase (+74.32% and +14.66%) according to longer-lasting battery capacity and higher doctor mobility, respectively. In contrast, the sojourn probabilities in context states of complementary sets, i.e., UC and LP Sets for battery charge, and Surgery, Open Air, and Home Sets for doctor's mobility, decrease. Moreover, it is worth noting how such variations disappear if we consider the broader Battery and Location Sets obtained by the union of (1) UC, LP, and HP Sets and (2) Patient's, Surgery, Open Air, and Home Sets, respectively. Indeed, both *leaving_patient's_home* and *discharging* can be considered internal actions to both Location and Battery Sets. The variation of their durations is then transparent to a coarser-grained analysis based on Location and Battery Sets. We also like to remark positive increments that occur for some single HOCEM variants, up to +2.31% when the average visit duration is set to 20 min (variants HOCEM 3, HOCEM 7, HOCEM 11).

Note that longer visits to patients may cause disconnection of the PDA from the power outlet and then an incomplete charging of the battery (see the dependencies among active and passive actions in Fig. 4) that can be charged only at doctor's home and at the surgery.

Moreover, we observe a larger impact of longer-lasting batteries in HP, LP, and UC Sets. Two additional hours over the given battery threshold increase the sojourn time in the HP Set up to +7.05% in HOCEM 4, and up to +12.69% in HOCEM 8 with four additional hours. Conversely, the PDA usage with a low charged battery (LP Set) and the need for a power outlet to charge the PDA (UC Set) decrease with more

powerful batteries (−4.73% in HOCEM 11 and −8.11% in HOCEM 8, respectively).

Finally, we observe a generic trend among HOCEM variants in HP, LP, and UC Sets. Given the same battery capacity (e.g., HOCEM 4, HOCEM 5, HOCEM 6, and HOCEM 7), the benefit of bigger batteries (i.e., longer sojourn times in HP Set) is proportionally reduced by longer stays at patients' homes.

The above considerations represent the typical results of an analysis that could not be obtained without a formally quantified approach to context modeling. This sensitivity analysis highlights the context parameters that have the highest impact on sojourn probabilities. As mentioned in Sect. 4.1, such an analysis allows results to be made more robust in the presence of inaccuracies on the estimated values of the parameters. Although these inaccuracies could be evidenced only by validating the results on a monitored system, such an analysis provides a support to designers when they have to evaluate alternative design choices.

5 Applying context modeling to non-functional analysis

In this section, we apply our context reasoning approach to software performance and reliability analysis.

Figure 9 depicts how the context reasoning workflow described in Sect. 4 can be integrated with two model-based analysis methodologies for performance [17,44] and reliability [18] analysis purposes. The two considered methodologies share similar input artifacts, i.e., a UML model made of several views and properly annotated for the specific analysis. An additional context view, made of the FOCEM models described in Sect. 4, is integrated into the UML model. Thanks to this integration step, original context-agnostic non-functional methodologies become context aware.

The next sections detail the steps and related artifacts depicted in the bottom side of Fig. 9. In particular, in Sect. 5.1 we describe the software application modeling, in Sect. 5.2.1 the reliability analysis, and in Sect. 5.2.2 the performance analysis.

5.1 Software application modeling

This section illustrates the *design model* of MeH and its integration with our context modeling approach.

We adopt UML [31] both as design notation and as hosting notation for the formalism introduced in Sect. 3, for the following reasons:

- UML includes the modeling of statecharts, through its UML StateMachines language unit, which nicely fits our FOCEM/HOCEM modeling needs, as illustrated in Figs. 4 and 5.
- UML is extensible through *profiles*. We can then annotate the stochastic parameters (i.e., rates and probabilities) on state transitions through *stereotypes* and *attributes*.
- UML supports modularity. Reusable model elements can be collected in *model libraries*, such as FOCEMs and HOCEMs that can thus be reused for other applications running in similar contexts.
- Several non-functional analysis approaches accept UML-based design models as input [6,38], thus facilitating the integration of such an analysis methodologies with

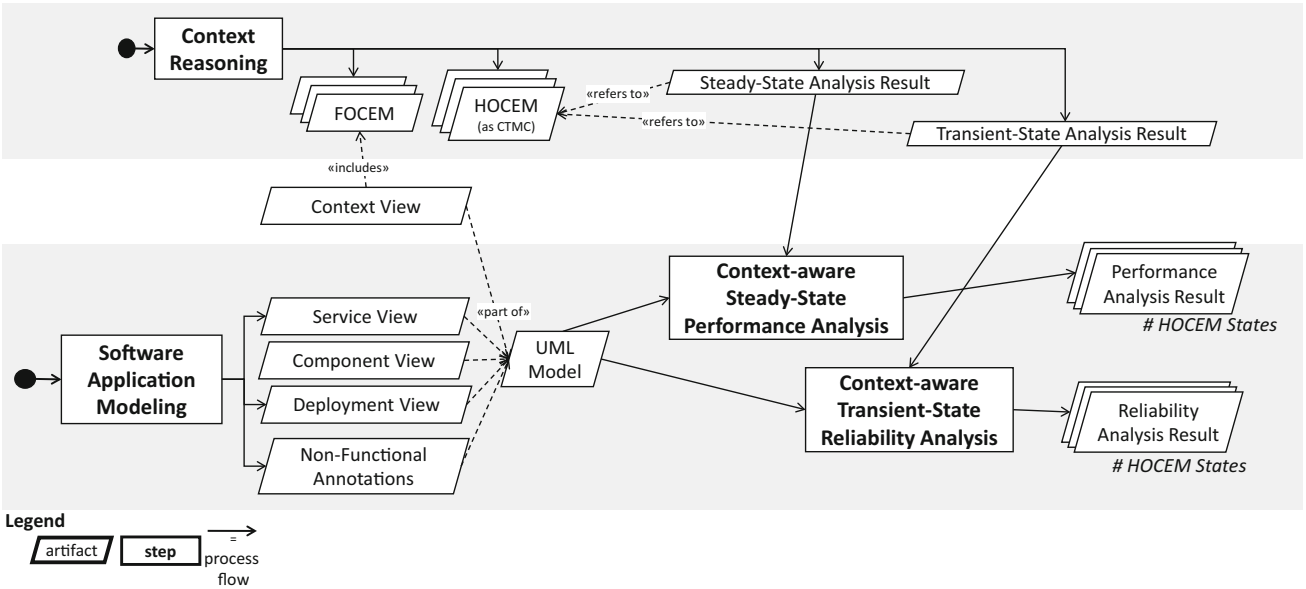


Fig. 9 Context reasoning applied to model-based performance and reliability analyses

our context modeling approach. We have exploited this opportunity by extending two existing reliability and performance analysis approaches in Sect. 5.2.

It is worth noting that our context modeling approach is constrained neither to UML, nor to any other modeling notation. Ad hoc domain-specific languages and notations can be devised for implementing the proposed approach through well-known MDE techniques (*metamodeling*) and tools (e.g., Eclipse Modeling Framework) [14]. However, this aspect is out of the scope of this paper and left as future work.

The design and context models represent the input to model-based non-functional analyses that will be illustrated in Sect. 5.2.

We have organized the UML model of MeH in views:

- A *Service View* (SV) represents the services provided by MeH as they are perceived and used by its external actors (Use Case Diagrams, UCD) along with their behavioral specifications (Sequence Diagrams, SD).
- A *Component View* (CV) illustrates the MeH software architecture in terms of its constituting software components and their provided/required interfaces (Component Diagrams, CD).
- A *Deployment View* (DV) models the allocation of software artifacts on the execution nodes. It also includes a representation of processing, communication, and storage resources of the supporting platform (Deployment Diagrams, DD).

In addition to such application views, as mentioned above a fourth crosscutting *Context View* embeds the UML StateMachines corresponding to the FOCEMs. A FOCEM can be associated with any UML modeling element from the Service, Component, and Deployment Views, whose attributes are part of the context sensed by the MeH application.

The Context View for MeH includes the FOCEMs shown in Fig. 4, and it combines with the other views as follows:

- A FOCEM is assigned to the Doctor actor (in the SV) to model the doctor mobility across different physical places.
- Two FOCEMs are assigned to the Battery and CPU nodes (in the DV) to model the evolution of charge–discharge cycles and execution modes, respectively.

The envisaged UML views require several profiles, namely

- The UML Standard Profile, which specifies a set of predefined standard *stereotypes* to identify executable artifacts (*Executable*) as manifestation of their logical counterpart in the software architecture (*manifest* relationship from components) deployed on execution hosts. These stereotypes are applied to diagrams in Figs. 11 and 13 [31].
- The Mobility Profile introduced in [25] and adopted to model logical and physical mobility and allocation of architectural elements of mobile systems. We adopt it to detail the physical mobility of doctors and their PDAs in Fig. 13.
- The MARTE (Modeling and Analysis of Real Time and Embedded systems) and its extension DAM (Dependability Analysis Model). In particular, MARTE [32] enables UML to support specification and analysis of non-functional properties (NFPs) in terms of performance attributes. Later on, the DAM [10] profile accomplished the same tasks for dependability attributes (including reliability). Indeed, DAM, being a MARTE specialization, can be used together with MARTE in UML models to jointly annotate performance and reliability properties, metrics, and input parameters.

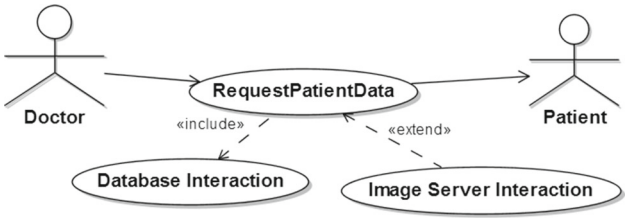


Fig. 10 The MeH use case diagram

In the following four subsections, the four views of the MeH system are separately described through a set of UML diagrams.[5]

5.1.1 MeH service view

The MeH application provides the RPD service to doctors for assisting their patients: a doctor equipped with a PDA is able, through a distributed MeH service, to retrieve mixed media information on patients, such as text with or without different kinds of images that refer to their personal data, their medical histories, and their diseases.

The RPD service is shown as a use case in Fig. 10 where the *Doctor* and *Patient* have, respectively, an active and a passive role. The RPD service is invoked by doctors, and it always includes the retrieval of textual information about the patients via interaction with a database. Moreover, depending on the current context, the same RPD service may be

[5] Here the UML diagrams have been suitably tailored to preserve their readability. However, they have been conceived to be machine readable by means of model transformations to fully support a model-driven approach. The complete UML model can be downloaded at https://code.google.com/a/eclipselabs.org/p/context-manager/.

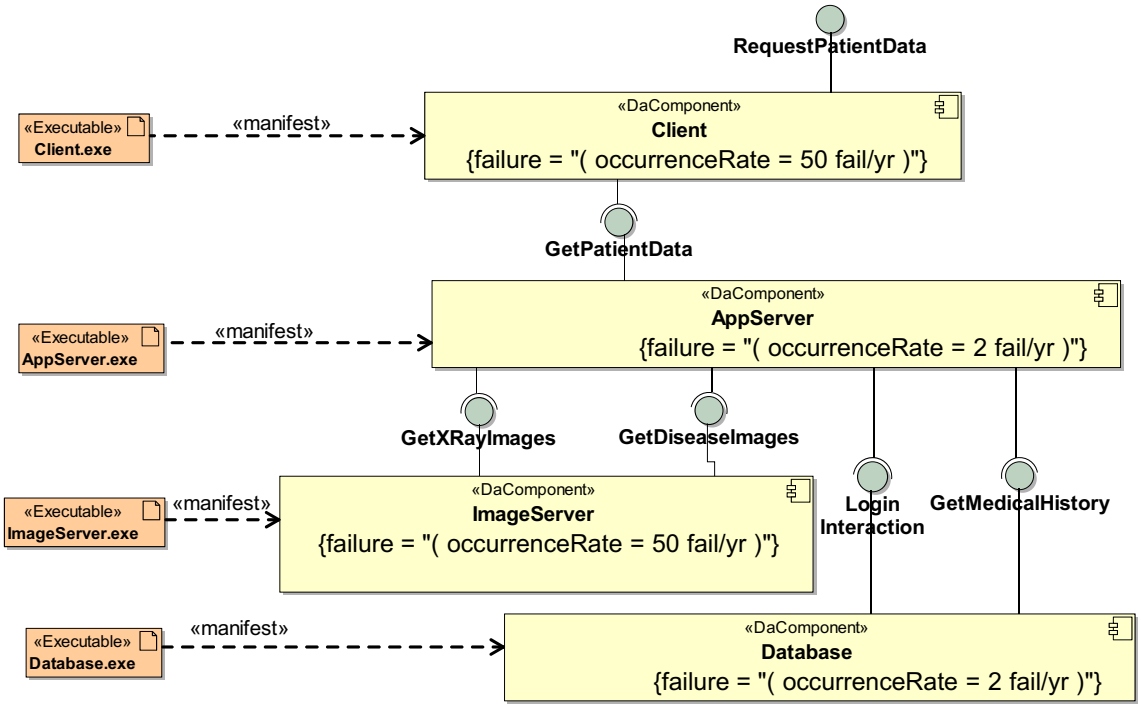


Fig. 11 The MeH software architecture

extended by an additional interaction with an image server that further detail patients' reports with images.

5.1.2 MeH component view

The CD in Fig. 11 shows the software architecture of the MeH application in terms of its component types (i.e., *Client*, *AppServer*, *Database*, and *ImageServer*), their connectors modeled as properly wired required/provided interfaces, and their executable artifacts. If not explicitly modeled, we assume a default multiplicity value of 1 for all software resource types.[6]

The view is completed by the specification of service behaviors. Figure 12 shows a Sequence Diagram associated with the RPD. When the doctor, once logged in, invokes the RPD service, the application server is in charge of retrieving data from a local database and, if needed, from an image server for patients' disease-related images (e.g., X-ray images). Finally, the result is displayed on the client. We envisage two alternative behaviors for RPD: (1) a *Standard Behavior* allows the retrieval of both text and images and is represented by the whole Sequence Diagram of Fig. 12; (2) a *Resource-Constrained Behavior* excludes the download of images and is represented by the Sequence Diagram of Fig. 12 without any interaction with *ImageServer*.

5.1.3 MeH deployment view

Figures 13 and 14 show the MeH hardware platforms at two different levels of detail.

Figure 13, which is inspired by the modeling solution introduced in [25], shows the overall system architecture that can be logically partitioned in two levels:

- The *Hosts Level* comprises the execution environments (*GaExecHost*) where the executable artifacts of the software components shown in Fig. 11 are deployed (e.g., Client.exe in PDA) and the communication happens (*GaCommHost*).
- The *Physical Locations Level* includes the places in which the execution and communication resources at the Hosts Level may reside, while the MeH system is executing. The *AllowedNodeLocation* stereotype identifies the possible places where an execution host can physically reside. According to the locations and mobility of doctors (see the FOCEM in Fig. 4a), this level includes four places for the PDA (i.e., Home, Surgery, Patient's Home, Open Air). For non-mobile execution hosts (i.e., AppServer Host, Database Host, ImageServer Host), we set a unique location (i.e., Server Room).

[6] We discuss the effect of multiplicities on non-functional analysis at the end of Sect. 5.1.5.

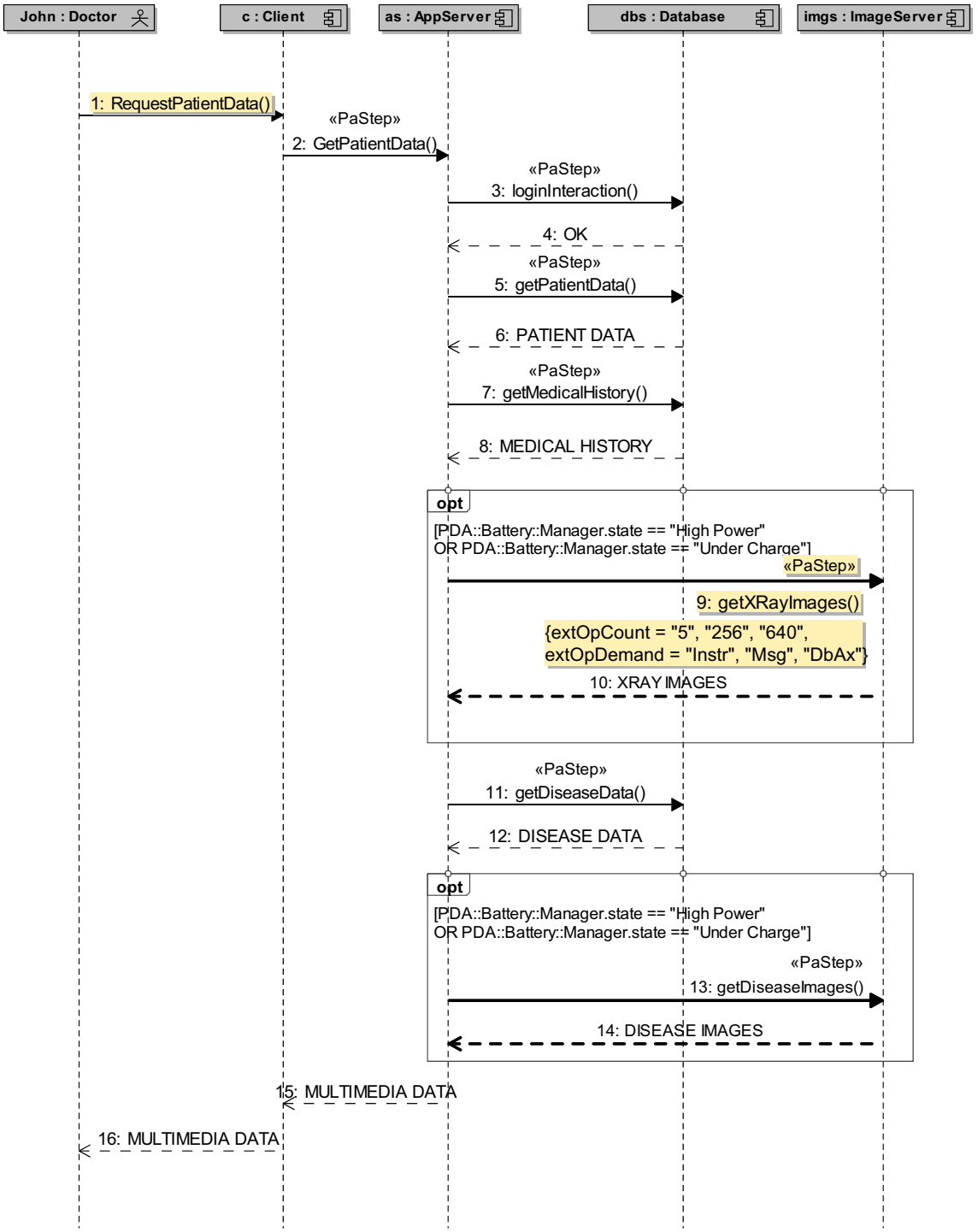


Fig. 12 The RPD sequence diagram

Places can provide different types of network connections that are modeled as typed ports on them: a 3G WAN is available in the Open Air, and wired LAN (e.g., 802.11g) is available both at doctor's Home and at the Surgery.

By reusing the nesting capability of UML Node (i.e., the base metaclass of the Place stereotype), we can also model (1) the containment relationships among places and (2) the sharing of certain resources. For example, the 3G network, which is available in the Open Air, is also available at doctor's and patient's homes as well as at the surgery. In contrast, LANs are only accessible at doctors' home (@Home) and

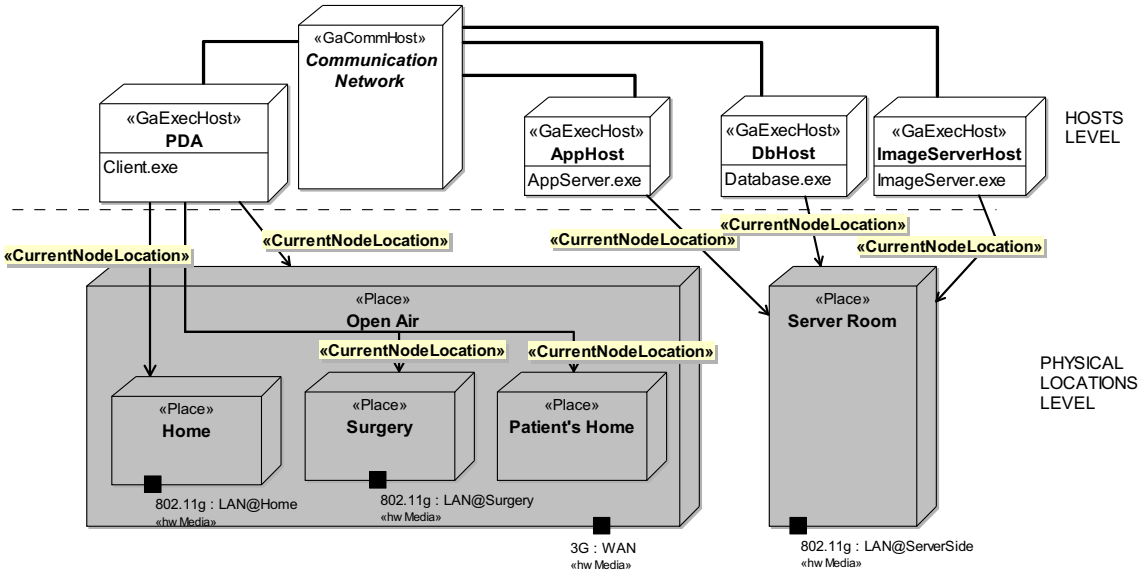


Fig. 13 The MeH dynamic deployment diagram

at the surgery (@Surgery). In the same manner, executables located in the server room can access a LAN (@ServerSide).

The inner hardware resources (CPU, Battery, Display, WiFi Card, 3G Card) of the PDA and the networks to which the PDA and the other execution hosts can connect (LAN@Home, LAN@Surgery, LAN@ServerSide, WAN) are shown in Fig. 14. For sake of readability, we omit to draw the inner details of AppServer Host, Database Host, and ImageServer Host nodes. However, similar to the PDA, all the execution nodes include (at least) processing, storage, and communication resources that are exploited by the software executables deployed on them.[7]

5.1.4 MeH context view

The UML model described so far is agnostic of the context evolution. A Context View has to be defined for context evolution modeling; hence, the notation-independent modeling approach described in Sect. 3 is here realized in UML. In Fig. 4, we have shown the FOCEMs for three different CAs: physical location of doctors, charge level of the PDA battery, and execution modes of the PDA CPU.

Through UML mechanisms, we associate each FOCEM with a UML element. In our case, the FOCEM in Fig. 4a is associated with the Doctor UML actor, whereas the ones in Fig. 4b, c with the Battery and CPU UML Deployment Nodes, respectively.

In addition to the UML StateMachines representing the FOCEMs, Context View also includes the modeling of the context-aware behaviors of the provided services.

Figure 15 shows an Interaction Overview Diagram for the RPD Service. A decision node precedes two boxes that refer to the Standard and Resource-Constrained alternative behaviors for RPD. Alternative behaviors are chosen according to a context condition. The latter may be expressed as a first-order logic proposition whose variables refer to the states of the FOCEMs.

For RPD, only the Resource-Constrained behavior (i.e., the one excluding the download of images) is available when the battery reaches a low-power state, in all other cases the Standard Behavior runs. Given the HOCEM in Fig. 5, this means that RPD is available in any context state, but with reduced capabilities in the states in the LP set.

It is worth noting that both the service behavioral specification (depicted on the SD in Fig. 12) and context behavioral alternatives (depicted on IOD in Fig. 15) are modeled via UML Interactions. It is up to the modeler to provide a coherent combined view. In this example, the same SD of Fig. 12 is referred to by both InteractionUses (i.e., the ref boxes) of the IOD in Fig. 15, and the same context conditions activate both alternative flows on IOD and opt fragments on the linked SD.

5.1.5 Adding non-functional annotations

Concerning the MARTE and DAM annotations, some model elements require the following additional information to derive the analysis models used in the approaches illustrated in Sect. 5.2. In particular:

[7] Note that the specification of these inner details makes hardware nodes as elements that can be saved in UML model libraries, so that they can be reused in other contexts or for other applications.

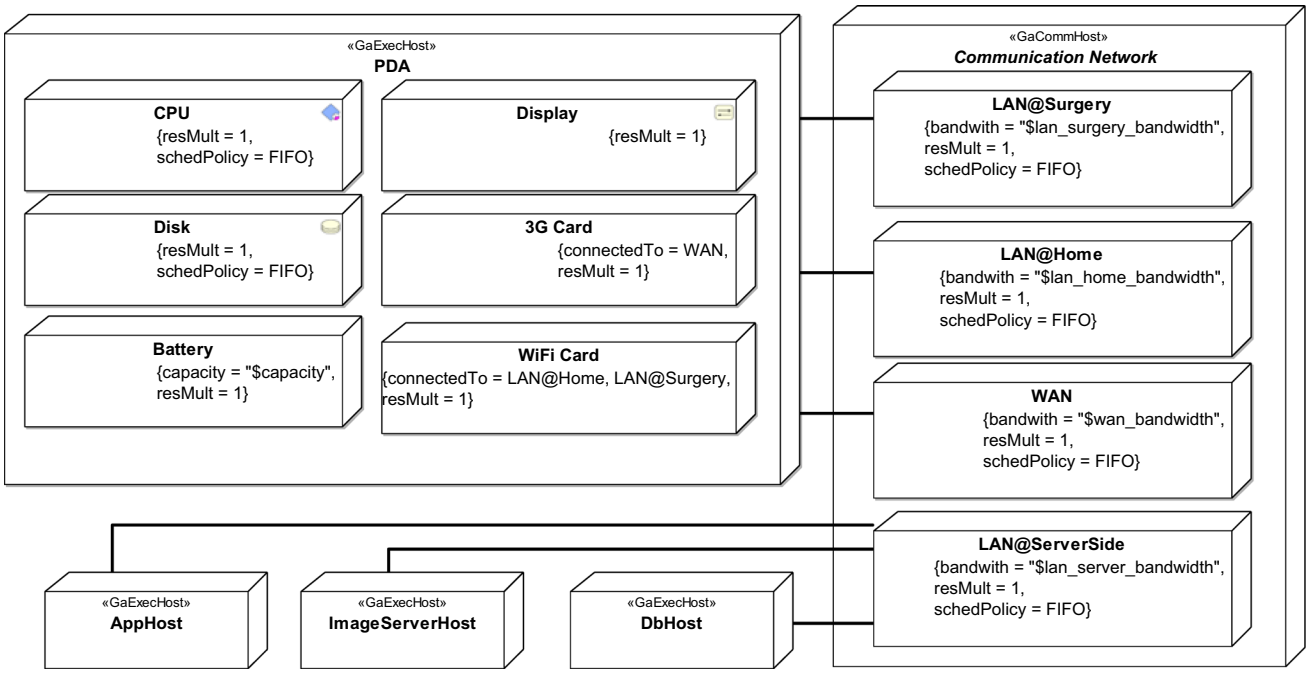


Fig. 14 The hardware platform specification of the doctor's PDA

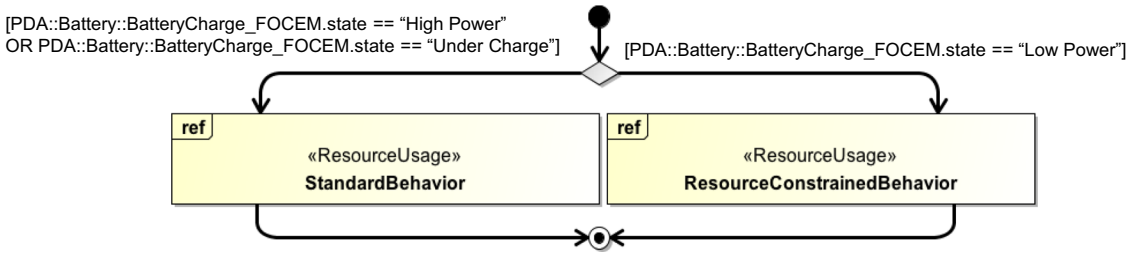


Fig. 15 Two different behavioral specifications for the RPD service

– The failure rate of each software component (DAM *DaComponent*, failure attribute) shown in Fig. 11 for the ImageServer component. This parameter is used to calculate the reliability of the whole MeH system.
– The *resource demand vectors* [44] for the messages exchanged among software components. A resource demand vector annotates the amounts of high-level or *logical resources* required to complete each execution step.[8] In particular:
  – Instr represents the number of high-level instructions to be executed from a CPU.
  – DbAx represents the number of mass memory blocks to be accessed on a disk.
  – Msg represents the number of bytes to be exchanged through a communication network.

Figure 12 shows a modeling solution through the MARTE profile. A getXRayImages() call message is annotated with the *PaStep* stereotype demanding for a certain amount (extOpCount attribute) of logical resource types (extOpDemand attribute) for its execution.
– The detailed characteristics of low-level or *concrete hardware resource* types for client and server hosts as well as communication networks. For this purpose, different detailed hardware resource configurations can be modeled via MARTE stereotypes (HwEndPoint, HwI/O, HwMedia, HwMemory, HwPowerSupply, and HwProcessor in Fig. 14) and variables ($-prefixed strings added as stereotype properties' values).[9] In particular, MARTE variables can act as placeholders for different input parameters like service times for CPU ($*cpu_pda_instr_service_time*), access times

[8] We assume that the resource units are implicitly released at the end of each step.

[9] The annotations for hardware resource configurations must be compliant with the Hardware Resource Modeling (HRM) subprofile and the Value Specification Language (VSL) included in [32].

Table 5 Detailed characteristics of the processing, storage, and communication resources for MeH

Execution host	Hw resource		Attribute (MARTE variable)	Value	Unit
PDA	CPU	1 GHz	$cpu_pda_instr_service_time	1.00E−09	s
	CPU	0,67 GHz	$cpu_pda_instr_service_time	1.49E−09	s
	SSD	0.1 ms	$disk_pda_access_time	1.00E−04	s
AppServer Host	CPU	3 GHz	$cpu_appserver_instr_service_time	3.33E−10	s
	Disk	9 ms	$disk_appserver_access_time	9.00E−03	s
Database Host	CPU	3 GHz	$cpu_database_instr_service_time	3.33E−10	s
	Disk	9 ms	$disk_database_access_time	9.00E−03	s
ImageServer Host	CPU	3 GHz	$cpu_image_server_instr_service_time	3.33E−10	s
	Disk	9 ms	$cpu_image_server_access_time	9.00E−03	s
WAN	WAN		$wan_bandwidth	4.27E+01	MB/s
LAN@Home	DSL 20 Mbps		$lan_home_bandwidth	8.00E−01	MB/s
LAN@Surgery	DSL 20 Mbps		$lan_surgery_bandwidth	8.00E−01	MB/s
LAN@Server	DSL 10 Gbps		$lan_server_bandwidth	1.56E−03	MB/s

for disks ($*disk_pda_access_time*), and bandwidths for communication networks (e.g., $*lan_surgery_bandwidth*). These parameters can be obtained from the clock speed, access time to memory units,[10] and bandwidth, respectively. The properties of the hardware resources, the corresponding MARTE variables, and the actual values of all these resources are listed in Table 5.

It is worth noting that we assume as undefined the multiplicity value for all software and hardware resource types (i.e., the maximum number of instances of the resource considered as available [32],[11]) if not otherwise modeled for non-functional analyses. For sake of illustration, we have assumed single multiplicity of components over all the case study considered in this paper. The introduction of multiple instances, however, does not affect the context modeling, whereas it may affect the complexity of the non-functional analysis, depending on the analysis approach adopted. In particular: (1) the reliability analysis adopted here would not be affected, because the model can deal with multiple instances, whereas (2) the performance analysis adopted here, as mentioned later, is not adequate for multiple instance models, because it does not consider resource contention. In order to address contention, more complex performance models, such as queueing networks, should be adopted.

[10] We consider memory units of 4 KB. For the solid state drive (SSD) the data access is set to 0.1 ms, while the hard disk drive (HDD) access time estimation derives from different parameters like average seek time, disk spins, transfer rate, controller overhead, and average rotational delay.

[11] In MARTE [32], the default value for resMult is 1. For simplicity, we do not introduce additional MARTE variables for software and hardware resource multiplicities to represent undefined values.

5.2 Context-aware non-functional analysis

In this section, we build up non-functional analysis on our context modeling approach. In particular, we tailor two approaches to reliability and performance analysis to work on models of context-aware applications built with our approach. This aims to show that not only does our modeling approach allow introducing all information necessary to perform non-functional analysis, but also it enables multiple types of analysis whose results are critical to support decisions in a context-aware domain.

We have adapted two existing approaches for reliability and performance analysis to the case of context-aware systems, as it will be detailed in the following. However, other state-based non-functional analyses can be adapted to our context approach, given that they require the same modeling information. Thereafter, we have used the reliability approach for a transient-state analysis, whereas the performance approach for a steady-state analysis. This choice (i.e., transient vs. steady state) has not been driven by the intrinsic characteristics of the approaches, as they can be easily applied to the other case, but by the intent of illustrating that both situations can be tackled with our context approach.

In general, the selection of a non-functional analysis approach depends on the non-functional system requirements. For example, the reliability approach that we use here does not consider error propagation (i.e., it assumes only single points of failures), whereas in a different type of system, such as one that adopts fault tolerant mechanisms because it faces more stringent reliability requirements, the error propagation plays a crucial role, so a different reliability modeling approach has to be adopted. Similarly, the decision whether transient-state or steady-state analysis is more appropriate depends on the system scenarios of interest, namely whether

a specific interval of time or the long-term average behavior is of critical interest, respectively.

The analysis that we present here is not aimed, however, at being adopted at runtime, where real-time requirements claim for lightweight approaches that can return quick results useful for online decisions. As we have mentioned before, this paper target is the system design phase, when multiple alternatives have to be considered and compared to each other in order to support design decisions.

5.2.1 Context-aware transient-state reliability analysis

Here we consider the reliability analysis approach introduced in [18] to study the reliability of a software architecture as a function of the reliability of its software components.

The failure probability FP of a system Sys at time $T < t$ can be expressed as:

$$FP_{Sys}(T < t) = 1 - \prod_{i \in C_{Sys}} FP_i(T > t) \tag{1}$$

where the probability that Sys fails before time t is expressed as the complement of the probability that all software components (composing set C_{Sys}) do not fail before t. If the component failures are exponentially distributed, then Eq. (1) becomes:

$$FP_{Sys}(T < t) = 1 - \prod_{i \in C_{Sys}} e^{-\lambda_i t} \tag{2}$$

where λ_i represents the failure rate of component i. We define the system failure probability in a specific context state s_j through Eq. (2) applied to the set C_{s_j} of components providing services in context state s_j, namely:

$$FP_{s_j}(t) = 1 - \prod_{i \in C_{s_j}} e^{-\lambda_i t} \tag{3}$$

where we have simplified the notation by omitting T.

In Table 6, we associate failure rates λ_i with the four MeH software components, as they appear in the Component Diagram of Fig. 11. Failure rates in Table 6 hold for both ScA and ScB. It is worth noting that the failure rates are invariant with respect to context states, while the number of software components in C_{s_j} that are involved in the provision of RPD service varies with context states.

We assume that the Standard Behavior is chosen in context states where the battery is in *high-power* (HP) or *under charge* (UC) states, and the Resource-Constrained Behavior is adopted in all other states. Such a contextual condition is annotated on the decision point of the UML Interaction Overview Diagram in Fig. 15. Consequently, the Image Server component is not involved in those context states where the Resource- Constrained Behavior is chosen, that is, when the battery is in the *low-power* (LP) state. As reported in Table 6, the Image Server component is the most unreliable one among the servers.

Starting from Eq. (3), the transient-phase failure probability of a context-aware software system, given a set S of possible context states ($s_j \in S$), can be obtained by summing the system failure probability in each context state s_j weighted by the corresponding sojourn probability $p_{s_j}(t)$ at time t, namely

$$FP_{Sys}(t, S) = \sum_{s_j \in S} p_{s_j}(t) \cdot FP_{s_j}(t) \tag{4}$$

where the values of $p_{s_j}(t)$ are shown in Figs. 6 and 7 for the HOCEMs of both ScA and ScB.

Equation (4) is calculated for the MeH system, and the results are shown in Fig. 16. Four distinct curves illustrate the variation of the MeH failure probability at different instants t during the transient phase.

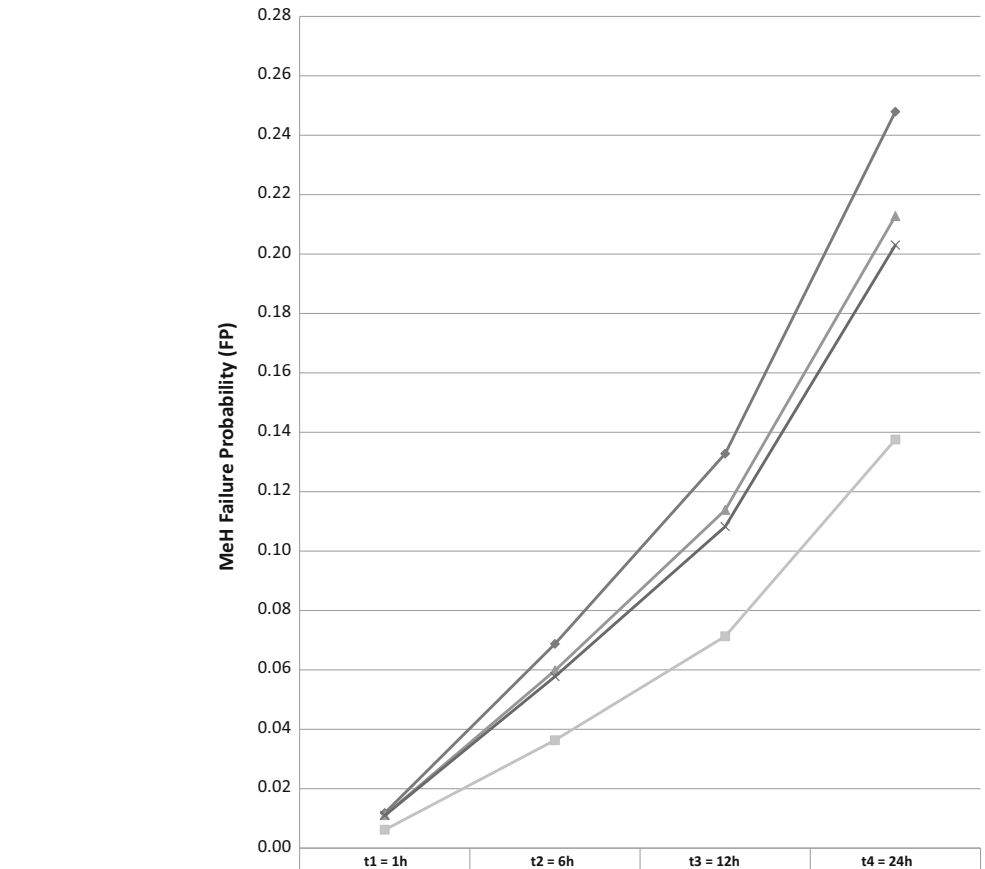
Standard Behavior only and *Resource-Constrained Behavior only* curves correspond to context-unaware cases, where the same specific behavior for the RPD service is provided in each context state. It is worth noting that such curves also represent the upper and lower bounds of the failure probability for MeH, respectively. Indeed, the Resource-Constrained Behavior is composed by a strict subset of the interactions required by the Standard Behavior (i.e., the former excludes any interaction with the unreliable Image Server). And since the failure probability of an inactive software component is ignored in Eq. (3), bounds are obvious.

The other two curves labeled as *Context-Aware Behavior* illustrate the reliability of the MeH system in the two context scenarios ScA and ScB. In such cases, the proper alternative behavior for the RPD service is chosen according to the context state with a certain probability at a given instant during the transient phase.

These curves appear in the figure very close, but this is due to the need for representing in the same figure also the bounding curves. In order to appreciate the actual differences of reliability between ScA and ScB, we have reported their numerical values along time on the bottom of Fig. 16.

Table 6 Reliability model parameters

Components	Client	App Server	Database System	Image Server
Failure rate (by year)	50	2	2	50



	t1 = 1h	t2 = 6h	t3 = 12h	t4 = 24h
Standard Behavior only	0.012	0.069	0.133	0.248
Resource Constrained Behavior only	0.006	0.036	0.071	0.138
Context-Aware Behavior (ScA)	0.011	0.060	0.114	0.213
Context-Aware Behavior (ScB)	0.011	0.058	0.108	0.203

Fig. 16 Transient-state reliability analysis for MeH

As expected, the failure probability in ScB is lower than in ScA along the whole timeline. Indeed, the latter scenario assumes a lower mobility of the RPD users, thus resulting in a generalized lower probability to sojourn in the LP Set with respect to ScB (−0.0879, see Table 3) and then to a lower probability to invoke the Standard Behavior of the RPD service that has a higher probability to fail due to the interaction with the Image Server component.

An interesting aspect of Fig. 16 is that the gap between the reliabilities of the mobility scenarios sensibly grows with time, because it depends on the different trends of transient-state probabilities in ScA and ScB, as illustrated in Figs. 6 and 7, respectively. In particular, as shown in Fig. 7, we observe that in the second part of the day the high mobility of RPD users brings them to locations, like Open Air ones, where their devices cannot be easily recharged, and thus, the Standard Behavior cannot be conveniently invoked. Hence, with time passing, ScB gains reliability with respect to ScA (of course, at the expense of missing patient analyses images, as outlined in other occasions before).

Several considerations can be inferred from these results. A first (simple) one is that the failures of the Image Server heavily affect the system reliability, and this is particularly evident across different mobility scenarios. Software designers are advised to spend more time on testing this server, by possibly analyzing also the influence of remote connection failures on its reliability.

A finer observation concerns the trade-off between the need of RPD by doctors to get patient's images and their tolerance to system failures. To investigate this aspect, designers can collect users' feedback about their perceived quality of service, in order to understand whether the Standard Behavior is appreciated at any time of the day from the users despite failures. If it is not, then the adaption strategy can be modified by providing, for example, in the hands of users the possibility to activate/deactivate the interactions with the Image Server, even when they are in contexts where their devices can be easily recharged. The study of alternatives like this one only implies, in our framework, to modify the context modeling and re-execute the reliability analysis.

Finally, we remark that, for sake of illustration, we have here considered a simple reliability model. Equation (1) assumes: (1) failure independence among components and (2) that a component failure corresponds to a system-level one (i.e., fail-and-stop mechanism). This equation can be easily solved with a basic worksheet, as it does not require complex solver tools. However, more complex reliability models are still compatible with our approach because the model complexity does not impact on the context modeling, whereas it obviously affects: the number of UML annotations, the complexity of the reliability model extraction from the UML model, and the reliability model solution. In such cases, more complex tools, such as SHARPE [47], can be adopted for reliability analysis.

5.2.2 Context-aware steady-state performance analysis

Here we consider the performance analysis approach introduced in [17]. In order to apply this approach, we have transformed the annotated MeH UML model described in Sect. 5.1 into an execution graph (EG) [44]. This is a platform-independent model that represents the software dynamics along with its requests of resources called *resource demand vectors*. The latter may be expressed, for example, in terms of processing, storage, and communication resource units (like virtual machine instructions, number of accesses to databases, and number of sent/received messages). Resource demand vectors are mapped to (more or less powerful) devices that provide hardware counterparts of such logical resources (e.g., CPU speed, disk access rate, and network bandwidth).

In Fig. 17, the EG of the RPD service is shown. It is obtained by combining the RPD sequence diagram in Fig. 12 and the RPD interaction overview diagram in Fig. 15.[12] The EG is partitioned in vertical swim-lanes, one for each lifeline in Fig. 12, that cut blocks into subsets. The topmost labels indicate the name of the software component executing each subset and the execution host on which such software component is deployed. EG blocks represent UML Call Messages, whereas context-based behavioral variations of Fig. 15 are represented by decision nodes that, in practice, avoid calls to ImageServer as prescribed for the Resource-Constrained RPD service variant, executed when the battery is in the *low-power* (LP) or *under charge* (UC) states.

A demand vector is associated with each EG block in light gray, and it is labeled as platform independent (PI). They are obtained from *GaAcqStep* stereotypes and properties as described in [17]. In particular, they report the amounts of resUnits and acqRes properties of *GaAcqStep* stereotype applied messages in Fig. 12. Note that each block included in both alternative RPD behaviors has two demand vectors associated, each reporting the amount of resources required in a specific behavior. In particular, two alternative demand vectors have to be assigned to Multimedia Data Upload and Multimedia Data Download blocks, representing the return of patient data sent from the server to the doctor's PDA, since their demands cannot be uniquely identified and depend on RPD behavior alternatives for the following reasons:

- The amount of bytes exchanged between AppServer and Client components depends on the presence and dimension of the downloaded images.
- The CPU instructions and the number of accesses to the Disk on the PDA increase with the need to process the (possibly) downloaded images.

The characteristics of platform resources of execution hosts are taken from the Deployment Diagram in Fig. 14, as listed in Table 5, and they are reported in Fig. 17 as dark gray vectors beside each EG block labeled as Platform.

The deployment of software components onto execution hosts drives the conversion of PI resource demands to platform-specific (PS) ones, and the conversion depends on the hardware resources equipping the target hosts. The same EG block can be executed on different platform configurations with different CPU clock frequencies (e.g., CPU with highest frequency set to 1 GHz in normal mode or to 667 MHz in power save mode) and different communication networks (DSL LAN at 20 Mbps at home or UMTS WAN at 384 Kbps in open air), depending on the particular context (see Fig. 5). Therefore, in Fig. 17 two platform configurations are associated with the RPD block. As a consequence, different PS demand vectors can be obtained from the same PI demand vector, as shown in Fig. 18 for the RPD invocation on PDA.

In PS vectors, resource demands are all expressed as service times.[13] We here assume that:

- For the Standard Behavior, the Client downloads 5 MBs of textual data and 100 MB of images.
- When downloaded, the images are displayed and further elaborated on the Client resulting in a processing overhead on the client CPU.
- When downloaded, all images are saved on the local Disk of the PDA, thereby causing several accesses.

[12] Note that the RPD interaction overview diagram considered for performance analysis is slightly different from the original one in Fig. 15 because the Resource-Constrained Behavior for the RPD service is here executed also when the battery is UC.

[13] The service time is the amount of time required by the particular resource (e.g., disk) to satisfy service requests (e.g., store data by accessing disk sectors).

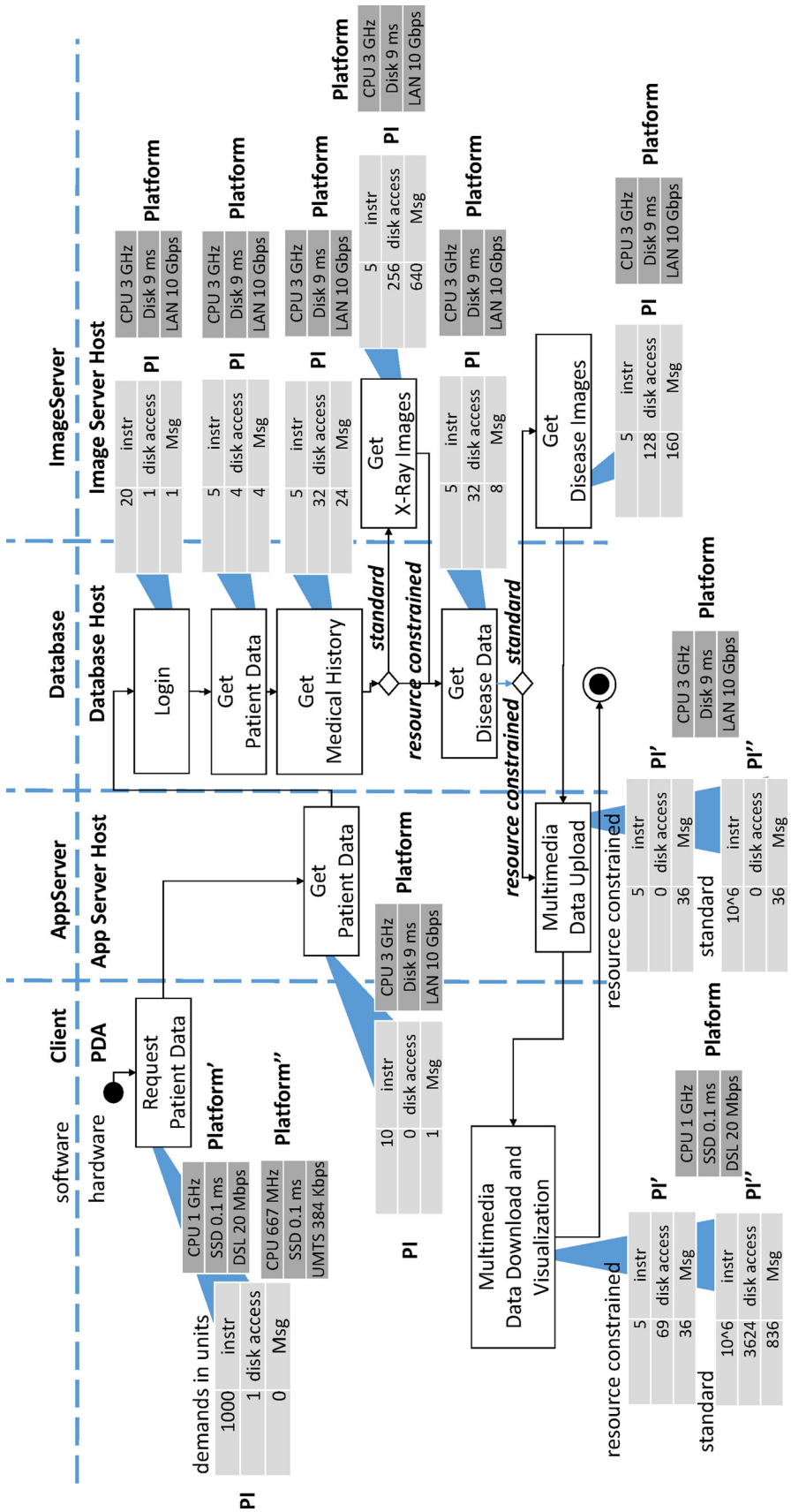


Fig. 17 EG of the RPD service

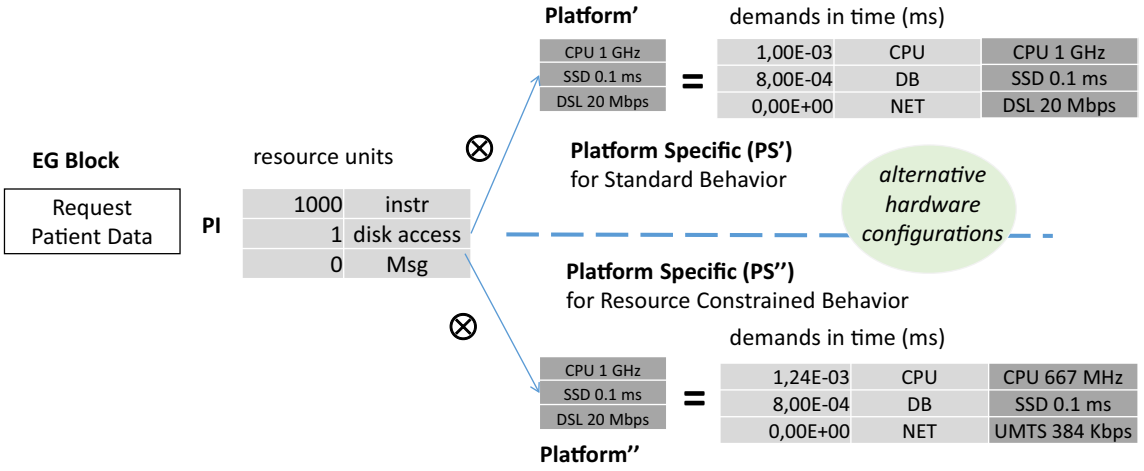


Fig. 18 Conversion from PI to PS demand vector

```
1  For each service s {
2    For each context state c in HOCEM {
3       identify the platform p adopted in context state c
4       identify the EG of the behavior b executed in context state c
5        for service s
6       For each block in this EG {
7          calculate the PS demand vector
8          calculate the service time of the block by summing
9           the PS vector entries
10      }
11      calculate the service time of behavior b by combining
12       the service times of its EG blocks
13   }
14 }
```

Fig. 19 Context-customized EG synthesis algorithm

We then apply the EG synthesis algorithm [44] to each of the 16 contexts identified in the HOCEM of Fig. 5 and sum up the service times of PS vectors to obtain context-specific performance indices for both Standard and Resource-Constrained RPD services by following the algorithm in Fig. 19.

For sake of illustration, in the following we stepwise apply the algorithm to the EG in Fig. 17. For this goal, the rows mentioned here below refer to the algorithm in Fig. 19.

For each context state, a context-specific platform is obtained in row 3 by selecting the CurrentNodeLocation relationships among the allowed ones in a Deployment Diagram (see MeH DD in Fig. 13).

Resource-specific service times of PS demand vectors are then carried out in row 7 for each EG block by multiplying the amount of each resource, as specified in its PI demand vector, by the service time of the corresponding resource in the platform listed in Table 5. The service time of each EG block is then obtained in row 8 as the sum of the resource-specific service times carried out in row 7.

For each possible execution flow on the EG, a flow service time is calculated in row 11 by combining the service times of its blocks. In the MeH system, we have calculated the service times of the two alternative execution flows of the RPD service, which correspond to the Standard and Resource-Constrained behaviors.

At the end of the inner loop (row 13), the service time of a service is obtained for each context state. At the end of the outer loop (row 14), the algorithm provides the service time of all system services in each context state, which for the MeH system is the RPD service only.

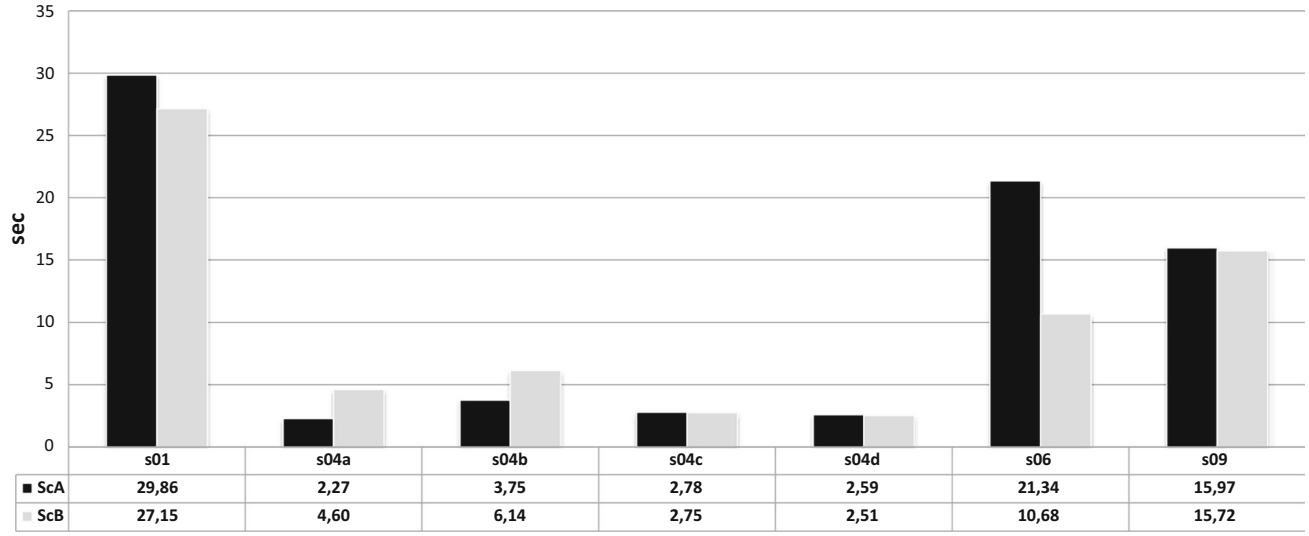
The outputs of this algorithm on MeH are two sets of 16 context-specific service times, one for each alternative behavior of the RPD service (i.e., Standard and Resource-Constrained). Table 7 illustrates these values.[14]

A maximum response time of 4559.32 s is obtained when the doctor stays outdoors (i.e., in the OpenAir Set) and at patient's home, the battery is in high power, and then all the

[14] Note that, since no contention of resources is considered in this performance analysis, the algorithm for EG synthesis has been simply in-house developed.

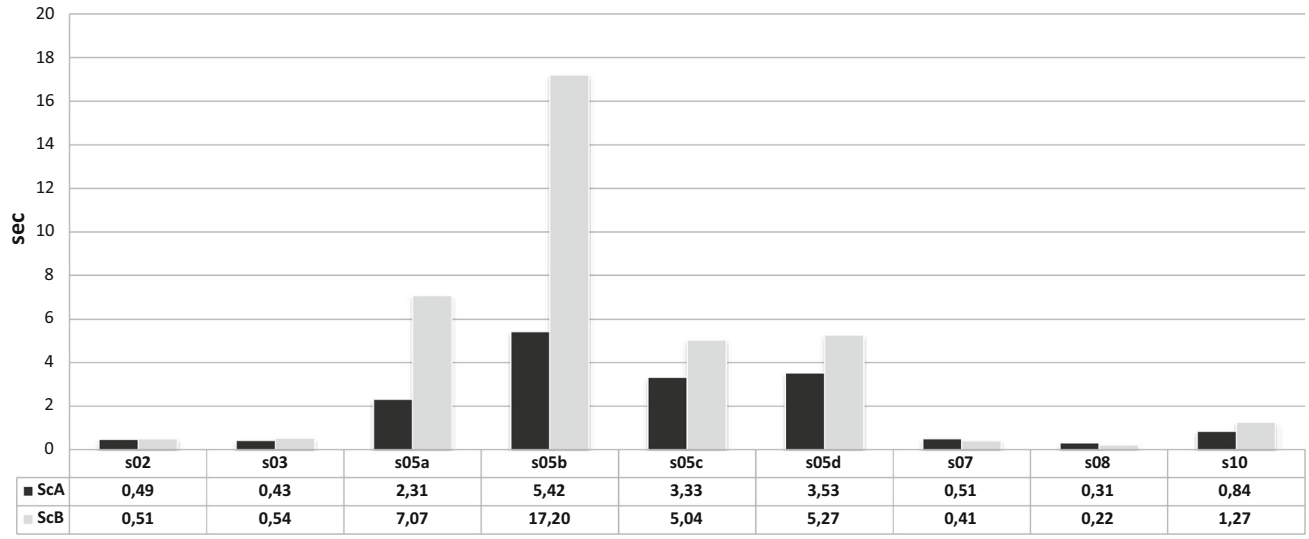
Table 7 Response times of the RPD service

Context state				RPD behavior	Deployment platform	Client-App Server Comm. Network	RT (s)	Min	Max
Name	Phy.	Batt.	CPU						
s01	H	HP	NM	Standard	LAN@Home	DSL 20 Mbps	184.25		
s02	H	LP	PSM	Resource Constrained	LAN@Home	DSL 20 Mbps	4.11	*	
s03	H	UC	PSM	Resource Constrained	LAN@Home	DSL 20 Mbps	4.11	*	
s04a	OAhs	HP	NM	Standard	WAN	UMTS	4559.3		*
s04b	OAsh	HP	NM	Standard	WAN	UMTS	4559.3		*
s04c	OAsp	HP	NM	Standard	WAN	UMTS	4559.3		*
s04d	OAps	HP	NM	Standard	WAN	UMTS	4559.3		*
s05a	OAhs	LP	PSM	Resource Constrained	WAN	UMTS	192.51		
s05b	OAsh	LP	PSM	Resource Constrained	WAN	UMTS	192.51		
s05c	OAsp	LP	PSM	Resource Constrained	WAN	UMTS	192.51		
s05d	OAps	LP	PSM	Resource Constrained	WAN	UMTS	192.51		
s06	S	HP	NM	Standard	LAN@Surgery	DSL 20 Mbps	184.25		
s07	S	LP	PSM	Resource Constrained	LAN@Surgery	DSL 20 Mbps	4.11	*	
s08	S	UC	PSM	Resource Constrained	LAN@Surgery	DSL 20 Mbps	4.11	*	
s09	P	HP	NM	Standard	WAN	UMTS	4559.3		*
s10	P	LP	PSM	Resource Constrained	WAN	UMTS	192.51		



	s01	s04a	s04b	s04c	s04d	s06	s09
ScA	29,86	2,27	3,75	2,78	2,59	21,34	15,97
ScB	27,15	4,60	6,14	2,75	2,51	10,68	15,72

Fig. 20 Contributions to the average response times of Standard RPD service—HP Set



	s02	s03	s05a	s05b	s05c	s05d	s07	s08	s10
ScA	0,49	0,43	2,31	5,42	3,33	3,53	0,51	0,31	0,84
ScB	0,51	0,54	7,07	17,20	5,04	5,27	0,41	0,22	1,27

Fig. 21 Contributions to the average response times of Resource-Constrained RPD service—UC and LP Sets

images are downloaded through the only network available, which is a WAN cellular network (e.g., UMTS).

A minimum response time of 4.11 s is obtained when the doctor invokes the Resource-Constrained version of the RPD service at home and at the surgery.

It is worth noting that the minimum and maximum values are invariant with respect to the scenarios ScA and ScB, because the underlying hardware platform remains the same.

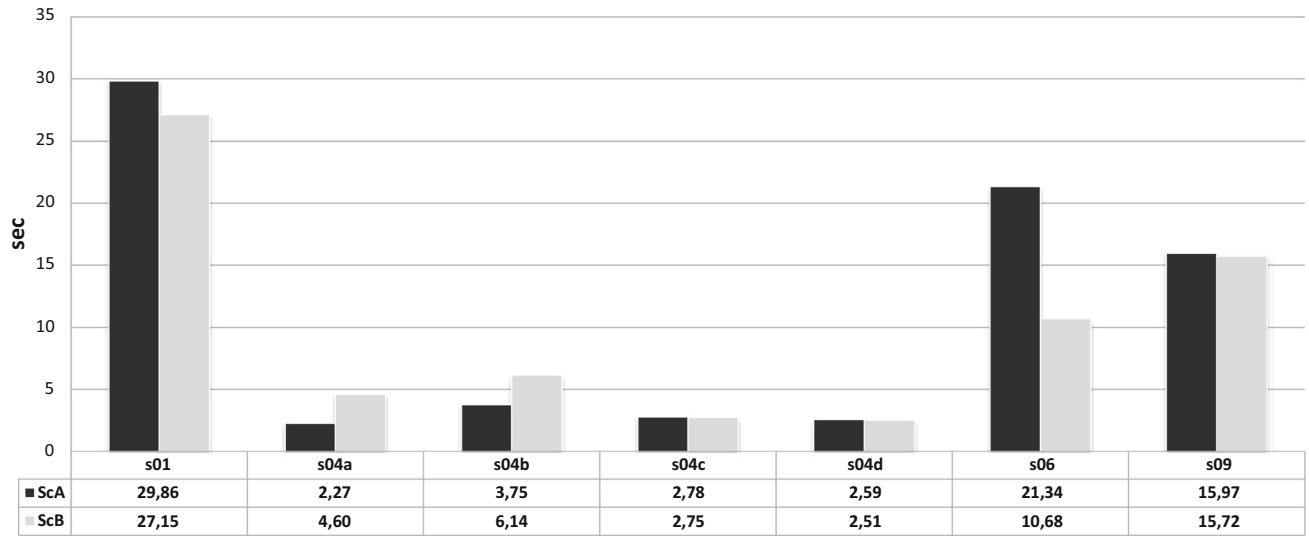
Then, we calculate the average service time of the MeH system in a steady state across the context states. The average service time is defined as the weighted sum of the service times of the RPD behaviors running in each context state, and it can be formulated as follows:

$$ST_{Sys} = \sum_{s_j \in S} \pi_{s_j} \cdot ST_{s_j} \tag{5}$$

where the weights π_{s_j} represent the steady-state probabilities calculated for each context state in Table 2.

The resulting average response time is 464.28 s in ScB and 349.98 s in ScA. Note that the former is quite higher than the latter mainly because in ScB the doctor moves more often than in ScA and therefore can experience low bandwidth networks.

In Fig. 20, we report the contribution to these average response times given by each context state where the Standard Behavior runs because the battery of the doctor's PDA is in high-power mode (HP). Similarly we do in Fig. 21 for the case of Resource-Constrained Behavior induced by battery



	s01	s04a	s04b	s04c	s04d	s06	s09
ScA	29,86	2,27	3,75	2,78	2,59	21,34	15,97
ScB	27,15	4,60	6,14	2,75	2,51	10,68	15,72

Fig. 22 New contributions to the average response times of standard RPD service—HP Set

in low-power mode (LP) or under charge (UC). As the histograms show, the highest contributions to the whole average response time across all states comes from s04a, s04b, s04c, and s04d that appear in Fig. 20 and that share the characteristic of using UMTS network connection, as listed in Table 7. This happens for both mobility scenarios ScA and ScB. Note that also in s09 the system runs the standard behavior in the presence of UMTS network connection, but, in this case, the contribution to the average response time is mitigated by the short stay of the doctor in that state in both scenarios, as listed in Table 2. These considerations highlight that the adaptation strategy is not fully adequate to guarantee good performance in all context states. This is because the adaptation strategy is not predicated on the network connection, but only on the PDA battery. However, this aspect would not have emerged without such a joint analysis.

Hence, in order to improve the performance of the system, the adaptation strategy should include also the condition of the network connection. To validate this result, we have imposed that the RPD service runs the standard behavior in all contexts where the battery is high and the network connection is not UMTS. By considering this new context-aware adaptation, the average response time for the RPD service decreases from 349.98 to 95.74 s in ScA, and from 464.28 to 107.07 s in ScB. In Fig. 22, we also report the new contributions to the average response time of the context states s01, s04a-s04d, s06, and s09, where the gain appears even more evident.

This analysis shows that the identification of performance degradation causes in context-aware software systems is often not trivial and that our approach provides an automated instrument, in the hands of software developers, aimed at quantifying and comparing performance indices in combination with contexts and adaptation strategies.

6 Related work

In [30], Hong et al. proposed a literature review and classification framework for publications related to context-aware systems. They identify five layered macro areas, namely (1) *Concept&Research*, (2) *Network Infrastructure*, (3) *Middleware*, (4) *Application*, and (5) *User Infrastructure*. The *Concept&Research* area, in particular, covers categories like overview, algorithm, development guideline, framework, context data management, evaluation, and privacy and security.

The framework that we have presented in this paper contributes to this area by proposing a context modeling and reasoning approach based on CEMs. We then have further combined the *Concept&Research* area with the one concerning model-based non-functional analyses of software systems. In this respect, our main contribution lies in the intersection of these two areas. Indeed, we started our research activities with the aim of adapting existing non-functional analyses methodologies (i.e., for performance [44] and for reliability [18]) that were adopted in previous works on context-agnostic software systems.

The next two sections illustrate the related work concerning context modeling and reasoning approaches (Sect. 6.1) and non-functional analyses applied to context-aware software systems (Sect. 6.2).

6.1 Context modeling and reasoning

Surveys on approaches in the *Concept&Research* [30] are provided in [46] and [13].

In [46], the approaches with respect to the data structures used for representing and exchanging contextual information are classified as follows:

- *Key-Value Models* represent the simplest data structure for context modeling.
- *Markup-Scheme Models* use a hierarchical data structure consisting of markup tags with attributes and content (e.g., XML).
- *Graphical Models* provide diagrams where context-related information can be shown in an ad hoc manner and/or annotated on preexisting shapes.
- *Object-Oriented Models* encapsulate context in reusable objects so that the access to context information and processing logic can be provided by well-defined interfaces.
- *Logic-Based Models* have a high degree of formality and represent context in terms of facts, expressions, and rules. A logic-based system is then used to manage the aforementioned terms and allows new facts to be added, updated, or removed. The inference process can be used to derive new facts based on existing rules in the systems.
- *Ontology-Based Models* are formal representations of knowledge as a set of concepts within a domain, and the relationships between those concepts.

A newer classification of context modeling and reasoning approaches appeared in [13], where the authors identified a set of requirements that a context modeling approach should satisfy.

Different from [46], in [13] context models are distinguished with respect to their level of abstraction, so that *high-level context model* includes *situational context* (e.g., *at home* or *at work*) that are derived by reasoning about low-level context data [22]. Situations are external semantic interpretations of low-level context data that inject meaning into the application so that possible adaptation actions, like the selection of alternative service behaviors, are triggered only when low-level context data correspond to situation changes.

Following the classification framework in [13], we have evaluated our context modeling and reasoning approach based on CEMs. A detailed description is reported in Appendix B. According to [13], our approach is capable of identifying *contextual situations*, since it provides a means for representing and composing different types of higher-level contexts or situations represented by state machines, as we did with our CEMs. Other approaches support the idea of modeling physical or logical places as states and movements as state transitions [20,24]. Our approach extends this usage of the state machine notation to possibly any kind of context information.

A general formal framework for modeling contexts as situations using graph-based models is presented in [22], where the context, its evolution, and relationships with context-aware systems are modeled through *situational graphs*, *context model*, and *fibrations* concepts. Our approach may be classified as a detailed and concrete instantiation of such generic concepts, in particular:

- A context evolution model is a kind of *situational graph*, further enriched with rates and probabilities on arcs for sake of context reasoning.
- A UML model represents a very detailed type of *context model* since, potentially, any model element with its attributes can describe part of the context.
- The combination of UML model with CEMs' states corresponds to *fibrations*, i.e., system configurations that in the proposed MeH case study correspond to alternative behavioral specifications available when predefined *if-context-is* conditions are verified.

However, different from our approach, no reasoning capabilities adopting fibrations models are proposed in [22].

In [1], Afanasov et al. present both a context-oriented design concept based on state machines and a corresponding programming model based on the nesC language, expressly conceived for resource-constrained platforms. In particular, their context design approach introduces two key concepts, *contexts* and *context groups*, which can be reasonably compared to our CEMs, FOCEM and HOCEM, respectively. Indeed, different from [1], we are not binding our approach to a specific programming model, but we extend context design capabilities by formalizing a stochastic extension of state-based context models (FOCEM and HOCEM), and its integration with UML models and model-based analysis methodologies [18,44].

Our idea of managing all context-related aspects with state machines can be also compared to solutions for the modeling and analysis of adaptable software systems where the notion of context was not explicitly identified as a first-class domain concept as in [28,32,48]. In [28], Uchitel et al. proposed the concept of *modes* to extend the Darwin architectural description language for modeling service-oriented computing systems. Modes are also language primitives in the Architecture&Analysis Description Language (AADL) [48] for modeling real-time and embedded systems. In both cases, they can be used to model the structural evolution of a software architecture at runtime. AADL, in particular, also allows the modal characterization of all its modeling elements (e.g., system, connectors, and properties). Thus, our logical and hardware FOCEM can be modeled as AADL component modes, whereas the overall HOCEM as system modes. In AADL, it is also possible to model the physical mobility by means of system modes. However, in this case, it cannot be associated with a system user as we do by associating the FOCEM to UML actors. More recently, the concept of

mode has been introduced in the context of cyber-physical systems [15] to take into account the level of uncertainty in the context change mechanism.

In this respect, we have adopted the concept of state-based notation for *modes* from [28,32,48] and broadened its application to represent heterogeneous context information. We have concretely generalized the concepts of modes by introducing CEMs. In addition, thanks to the extensibility and widespread adoption of UML, our approach can be (1) sized for different definitions of context and (2) used as a general modal-based modeling approach for context-aware software systems in multiple domains.

In [39], the authors propose a modeling approach and a visual modeling notation for context-aware social collaboration processes. Here, the context includes document and people information. This work shares with ours the idea of modeling and formalizing context through a state-machine-like notation. However, different from our approach, [39] aims at providing a context model for a specific application domain (i.e., context-aware social collaboration business processes) that is merged with the software specification in [39].

In [36], the authors describe a data collection campaign that shows how actual values of CAs can be collected from user-centric CSs and how such data allow the calculation of sojourn probabilities in predefined context states to be used for further analysis steps. In this respect, our context modeling and reasoning approach may be used to formalize such monitored data as distinct FOCEMs and combine them in different HOCEMs for sake of alternative and/or refined context reasoning results (i.e., sojourn times in different FOCEMs and HOCEMs). This work shows how to exploit context reasoning results for context-aware non-functional analyses.

6.2 On combining context reasoning and model-based analysis

The problem of representing and reasoning on context may be tackled in isolation, as in the work surveyed in [46] and [13], or in combination with other issues (e.g., the design and development of context-aware middleware) that shift the focus on other context-related research areas [30].

In this paper, we have combined context modeling and reasoning activities with non-functional analyses of software systems. We explicitly model and analyze context through CEMs whose steady- and transient-state probability vectors have been reused as a new additional input to existing methodologies for non-functional analyses.

In this respect, a comparable framework, namely Context-aware Quality Model Driven Architecture (CQ-MDA), was proposed in [2]. It is an extension of MDA [37] that can be used for quality control in pervasive computing environments. Software architecture, hardware platform, and context are explicitly defined within a modular metamodel, ContextualArchRQMM. Two different metrics, namely Time Behavior Metric (TBM) and Minimum architectural Adaptive Cost (MaAC), are calculated on context-specific platform-independent models (CPIM) through simulation and assuming different user workloads. CQ-MDA stresses the need for separation of concerns among context, software architecture, and hardware platform. This approach, however, seems to be lacking a clear semantics for ContextualArchRQMM. Therefore, it is unclear whether and how both context reasoning and simulation can be directly carried out on CPIMs and the proposed quality metrics obtained. In our approach, we have chosen to reuse UML without creating any new language with the potential benefit from the reuse and extension of UML-based approaches for the design and analysis of software systems.

To the best of our knowledge, [2] is the only one focusing only on the modeling and reasoning on context in combination with the non-functional analysis of software systems. Indeed, the research focus is usually shifted on software adaptation as effect of context analysis results. The work presented by Grassi et al. in [23] is representative of this research area. They proposed a modeling framework for QoS-aware self-adaptive software applications that presents some similarities with our work.

The framework in [23] is based on the definition of an intermediate pivot language, called D-KLAPER, aimed at providing instruments to transform software models into non-functional ones and analyze QoS characteristics when changes occur in the application and/or its execution context. This approach includes the generation of Markov reward models to analyze non-functional properties even in non-steady states of the system, as we have done for reliability analysis. Context evolution is then modeled as sets of trigger events. However, no state machines are provided relating these triggers to context state and transitions, as we have done with CEMs. Consequently, no dependencies among triggering events can be modeled in D-KLAPER while we have modeled them through remote firings among CEMs. However, the approach in [23] presents some advantages, such as the explicit representation of adaptation actions and the analysis of non-functional costs of such actions.

Modeling and reasoning on context tasks have been usually tackled by international projects like [21,35].

In [35], Eliassen et al. proposed a combined resource and context model using the OMG General Resource Model [3]. The defined model is included in the MUSIC middleware [43] that monitors the context and the resources to catch their changes and adapts the application to fulfill the users' QoS requirements. The approach is based on QoS predictors and utility functions. The adaptation is based on the concept of service plan [40], which is a platform-independent specifi-

cation containing information on service configurations, its dependencies on the environment, and its QoS characteristics. The work in MUSIC is mostly aimed at providing primitives and functions supporting self-adaptation in ubiquitous and service-oriented environments.

As in the MUSIC project, we have adopted UML profiles (MARTE [32] and DAM [10]) to annotate parameters to be used for performance and reliability analyses. Concerning the context-related attributes, they are annotated on UML structural elements like Classes and Ports without any additional information about their evolution. Differently, our state-based approach emphasizes the evolutionary nature of context and allows a higher degree of flexibility and modularity in modeling with respect to MUSIC. The dynamic, heterogeneous, and hierarchical nature of context information is modeled through separated and hierarchical sets of CEMs. Our approach allows a higher degree of flexibility and modularity while designing context (1) through the possibility of directly associating CEMs with different UML modeling elements (i.e., Actors, Components, and Nodes) and (2) by combining them through remote firing dependencies, if required.

Finally, MUSIC middleware does not provide any support to model and analyze non-functional properties of such systems before their implementation and deployment. In this respect, it would be possible, with our framework, the generation of service plans and the provision of QoS models that work as predictors of context-specific reliability and performance indexes.

Another interesting project is DiVA [21], which aims at providing an integrated framework for managing dynamic variability in adaptive systems. Different from other approaches where the dynamic adaptation is handled at code level, DiVA exploits both model-driven and aspect-oriented technologies to define an architectural model (including base, variant, and adaptation models) at design time. The composition and validation at runtime of alternative models allow: (1) the choice of the system configuration that best adapts to the changed execution context and (2) the deployment and execution of the chosen configuration supported by a reflective middleware [41]. However, this approach does not provide any support for non-functional analysis.

7 Discussion and conclusions

In this paper, we have introduced an approach to model multidimensional software contexts, which can deal with any type of context that can be modeled with a set of state machines. Being based on a clear separation of concerns between software and context modeling, our approach is independent of the application domain. Here, we consider the eHealth domain as a case study, but we are currently investigating its usage in the cyber-physical production system domain.[15] The semantic formalization of a context composition operator has provided a formal instrument for context-dependent software analysis. We have shown how context-aware steady-state performance analysis and transient-state reliability analysis can be carried out by combining our approach with existing non-functional model generation techniques. This combination has demonstrated that our modeling approach (1) allows introducing all information necessary to perform non-functional analysis and (2) enables multiple types of analysis whose results effectively support the decision process in the development of context-aware systems.

Moreover, non-functional analysis combined with context analysis reveals critical aspects and design errors that may not emerge without such a joint analysis, thus confirming that the identification of software quality degradation causes in context-aware software systems is often not trivial. Hence, our approach provides an automated instrument, in the hands of software developers, aimed at quantifying and comparing characteristics of software systems in combination with contexts and adaptation strategies.

The validity of our approach relies on the main assumption that context and its evolution can be represented as a Markov chain. In particular, the Markov property is required for reasoning on the context. Although we have uniformly applied our approach to the modeling of physical mobility and hardware configuration awareness (i.e., two CAs of very different nature), the Markovian property assumption turns out to be more or less strict depending on the particular kind of awareness. For example, the memoryless property does not perfectly match on the transition of a full battery to an empty battery, since the longer the battery is in the high-power state, the more likely is that it moves to the low-power one. The property works instead very well for modeling the physical movements of users from one place to another, with the exception of predefined paths (e.g., for a traveling salesman). The extent to which the Markov property assumption influences the accuracy of our approach in predicting the whole context evolution should be evaluated case by case, depending on the particular type of context that best fits the software application. In any case, it is worth reminding that suitable combinations of exponential distributions can approximate arbitrarily well basically any distribution.

In order to show how the context analysis results may vary upon variations of context parameters, we carried out a sensitivity analysis to discuss expected analysis results while changing the parameterization of FOCEMs and resulting HOCEMs.

In order to reduce context modeling errors, automated procedures can be introduced to synthesize FOCEMs from

[15] http://me-at-big.blogspot.co.at/2016/07/context-modeling-and-analysis-of-cyber.html.

sensing data. Such procedures sense the context variables of interest and, by analyzing the stored data, can produce the relative FOCEM. An example of such a procedure can be found in [9], where a prototype tool is proposed to collect data from a RESTful web service with the aim of publishing and then retrieving battery consumption data and thus very close to build a Battery Charge FOCEM.

Since our approach does not allow CEMs to change state during the execution of a service, but only between one invocation and another, as a short-term future direction we intend to introduce this characteristic in our framework without any heavyweight extension of the UML metamodel.

Besides, we intend to address more complex forms of adaptation that, for example, completely replace the internal structure and behavior of a certain component when needed [33].

Another future direction is the application of our approach to more complex performance and reliability analyses (e.g., performance under resource contention, reliability under error propagation) and/or to other non-functional attributes (such as availability and security).

Finally, an interesting direction could be to evaluate the proposed approach w.r.t. other similar approaches in terms of design guidance and quality of results.

Acknowledgements Open access funding provided by TU Wien (TUW). We would like to thank the reviewers for their insightful comments on the paper that led us to improve our work.

Appendix A: Case study detailed data

In Tables 8 and 9, we report the state transition matrices of the HOCEMs for ScA and ScB.

Rows and columns have been partitioned to identify three subsets:

- HP Set identifies a powerful context where the RPD service is always running on a fully charged PDA and the only CA that varies is the physical location of the doctor equipped with the PDA.
- LP Set, similar to the HP Set, identifies a set of resource constrained contexts where the RPD service is always running on a PDA where the charge level of the included battery is always equal to or lower than the chosen threshold, in our example 25%.
- UC Set includes the two context states that correspond to the invocation of the RPD service while the battery is under charge. According to the remote firing dependencies among the transitions of the Doctor Physical Location and the Battery Hardware Configuration FOCEM, the recharge happens only at the doctor's home and at the surgery. When the doctor leaves these two places to go home or to a patient's home, the recharge is interrupted (i.e., the PDA is unplugged from the socket plug) and, according to the charge level of the battery, a new context state belonging to the HP or LP Sets is reached.

In Tables 10 and 11, several transient-state probabilities vectors are shown as they are obtained by setting different execution times t (expressed in hours) of the MeH system.

Table 8 Transition rates of the HOCEM for ScA

source	target				s01	s04a	s04b	s04c	s04d	s06	s09	s02	s05a	s05b	s05c	s05d	s07	s10	s03	s08
	rate			doc	H	OAhs	OAsh	OAsp	OAps	S	P	H	OAhs	OAsh	OAsp	OAps	S	P	H	S
				batt	HP	HP	HP	HP	HP	HP	HP	LP	LP	LP	LP	LP	LP	LP	UC	UC
				cpu	NM	NM	NM	NM	NM	NM	NM	PSM	PSM	PSM	PSM	PSM	PSM	PSM	PSM	PSM
	doc	batt	cpu		Doctor's physical mobility with powerful battery							only battery consumption then mode change in CPU								
s01	H	HP	NM			0.100						0.177								
s04a	OAhs	HP	NM							2.000			0.177							
s04b	OAsh	HP	NM		1.000									0.177						
s04c	OAsp	HP	NM								3.000				0.177					
s04d	OAps	HP	NM							3.000						0.177				
s06	S	HP	NM				0.125	0.250									0.177			
s09	P	HP	NM						12.000									0.177		
	doc	batt	cpu									Doctor's physical mobility with limited battery							Start recharging	
s02	H	LP	PSM										0.100						0.533	
s05a	OAhs	LP	PSM														2.000			
s05b	OAsh	LP	PSM									1.000								
s05c	OAsp	LP	PSM															3.000		
s05d	OAps	LP	PSM														3.000			
s07	S	LP	PSM											0.125	0.250					0.533
s10	P	LP	PSM													12.000				
	doc	batt	cpu		Stop Recharging (Battery Full Charged) then mode change in CPU							Stop Recharging (Battery Level < Low Level Treshold)								
s03	H	UC	PSM		0.500								0.100							
s08	S	UC	PSM							0.500				0.125	0.250					

Table 9 Transition rates of the HOCEM for ScB

source	target				s01	s04a	s04b	s04c	s04d	s06	s09	s02	s05a	s05b	s05c	s05d	s07	s10	s03	s08
	rate			doc	H	OAhs	OAsh	OAsp	OAps	S	P	H	OAhs	OAsh	OAsp	OAps	S	P	H	S
				batt	HP	HP	HP	HP	HP	HP	HP	LP	LP	LP	LP	LP	LP	LP	UC	UC
				cpu	NM	NM	NM	NM	NM	NM	NM	PSM	PSM	PSM	PSM	PSM	PSM	PSM	PSM	PSM
	doc	batt	cpu		Doctor's physical mobility with powerful battery							only battery consumption then mode change in CPU								
s01	H	HP	NM			0.125						0.220								
s04a	OAhs	HP	NM							1.000			0.220							
s04b	OAsh	HP	NM		0.500									0.220						
s04c	OAsp	HP	NM								3.000				0.220					
s04d	OAps	HP	NM							3.000						0.220				
s06	S	HP	NM				0.250	0.500									0.220			
s09	P	HP	NM						12.000									0.220		
	doc	batt	cpu									Doctor's physical mobility with limited battery							Start recharging	
s02	H	LP	PSM										0.125						0.660	
s05a	OAhs	LP	PSM														1.000			
s05b	OAsh	LP	PSM									0.500								
s05c	OAsp	LP	PSM															3.000		
s05d	OAps	LP	PSM														3.000			
s07	S	LP	PSM											0.250	0.500					0.660
s10	P	LP	PSM													12.000				
	doc	batt	cpu		Stop Recharging (Battery Full Charged) then mode change in CPU							Stop Recharging (Battery Level < Low Level Treshold)								
s03	H	UC	PSM		0.500								0.125							
s08	S	UC	PSM							0.500				0.250	0.500					

Table 10 Transient-state probability vectors for ScA

Context state				Transient-state prob. vector ScB				
No.	Phy	Batt.	CPU	1	6	12	24	Steady (infinity)
01	H	HP	NM	0.7636	0.3633	0.2839	0.2585	0.2567
02	H	LP	PSM	0.1132	0.1348	0.1205	0.1165	0.1162
03	H	UC	PSM	0.0287	0.1214	0.1092	0.1037	0.1033
4a	OAhs	HP	NM	0.0341	0.0173	0.0132	0.0119	0.0118
4b	OAsh	HP	NM	0.0016	0.0141	0.0179	0.0194	0.0195
4c	OAsp	HP	NM	0.0021	0.0109	0.0134	0.0144	0.0144
4d	OAps	HP	NM	0.0001	0.0099	0.0124	0.0134	0.0134
5a	OAhs	LP	PSM	0.0065	0.0145	0.0127	0.0121	0.0120
5b	OAsh	LP	PSM	0.0003	0.0158	0.0251	0.0279	0.0281
5c	OAsp	LP	PSM	0.0004	0.0109	0.0158	0.0172	0.0173
5d	OAps	LP	PSM	0.0002	0.0109	0.0165	0.0182	0.0183
06	S	HP	NM	0.0406	0.1413	0.1712	0.1826	0.1835
07	S	LP	PSM	0.0068	0.0856	0.1137	0.1221	0.1227
08	S	UC	PSM	0.0010	0.0437	0.0673	0.0743	0.0748
09	P	HP	NM	0.0005	0.0027	0.0033	0.0035	0.0036
10	P	LP	PSM	0.0001	0.0027	0.0040	0.0044	0.0044
Total:				1.0000	1.0000	1.0000	1.0000	1.0000

Table 11 Transient-state probability vectors for ScB

Context state				Transient-state prob. vector ScB				
No.	Phy	Batt.	CPU	1	6	12	24	Steady (infinity)
01	H	HP	NM	0.7157	0.3023	0.2402	0.2334	0.23339
02	H	LP	PSM	0.1265	0.1210	0.1208	0.1223	0.12231
03	H	UC	PSM	0.0407	0.1324	0.1269	0.1291	0.12916
4a	OAhs	HP	NM	0.0592	0.0338	0.0250	0.0239	0.02391
4b	OAsh	HP	NM	0.0024	0.0295	0.0327	0.0319	0.03188
4c	OAsp	HP	NM	0.0028	0.0143	0.0146	0.0143	0.01426
4d	OAps	HP	NM	0.0012	0.0129	0.0134	0.0131	0.01304
5a	OAhs	LP	PSM	0.0143	0.0406	0.0365	0.0367	0.03669
5b	OAsh	LP	PSM	0.0006	0.0510	0.0849	0.0893	0.08933
5c	OAsp	LP	PSM	0.0007	0.0203	0.0257	0.0261	0.02615
5d	OAps	LP	PSM	0.0003	0.0203	0.0269	0.0274	0.02736
06	S	HP	NM	0.0282	0.0930	0.0939	0.0919	0.09182
07	S	LP	PSM	0.0059	0.0813	0.0974	0.0986	0.09856
08	S	UC	PSM	0.0009	0.0387	0.0511	0.0520	0.05204
09	P	HP	NM	0.0006	0.0035	0.0036	0.0035	0.00350
10	P	LP	PSM	0.0001	0.0051	0.0065	0.0066	0.00660
Total:				1.0000	1.0000	1.0000	1.0000	1.0000

Appendix B: Characterization of our approach within a context modeling classification

In [13], a set of requirements that need to be taken into account when modeling context information has been introduced, as reported in the following.

– *Heterogeneity and mobility.* Context information models have to deal with a large variety of context information sources that differ in their update rate and their semantic level. Some context information is sensed or derived from existing context information. A context model should be able to express those different types of context information, and the context management system should provide

management of the information depending on its type. Many context-aware applications are also mobile (i.e., running on a mobile device) or depend on mobile context information sources (e.g., mobile sensors). This adds to the problem of heterogeneity, as the context information provisioning must be adaptable to the changing environment. In addition, location and spatial layout of the context information play important roles due to this requirement.

- *Relationships and dependencies.* There exist various relationships between types of context information that have to be captured to ensure correct behavior of the applications. One such relationship is dependency whereby context information entities/facts may depend on other context information entities. For example, a change to the value of one property (e.g., network bandwidth) may affect the values of other properties (e.g., remaining battery power).
- *Timeliness.* Context-aware applications may need access to past states and future states (prognosis). Therefore, timeliness (context histories) is another feature of context information that needs to be captured by context models and managed by the context management system. The management of context histories is difficult if the number of updates is very high. It may not be feasible to store every value for future access, and therefore, summarization techniques need to be applied (e.g., the aggregation of position updates to a movement function using interpolation techniques, or the use of historical synopses of data).
- *Imperfection.* Due to its dynamic and heterogeneous nature, context information may be of variable quality. In fact, it may even be incorrect. Most sensors feature an inherent inaccuracy (e.g., a few meters for GPS positions), and the sensed values age if the physical world changes, so that this inaccuracy increases over time. In addition, the context information may be incomplete or conflicting with other context information. Thus, a good context modeling approach must include modeling of context information quality to support reasoning about context.
- *Reasoning.* Context-aware applications use context information to evaluate whether there is a change to the user and/or computing environment context; taking a decision whether any adaptation to that change is necessary often requires reasoning capabilities. It is therefore important that the context modeling techniques are able to support both consistency verification of the model and context reasoning techniques. The latter can be used to derive new context facts from existing context facts and/or reason about high-level context abstractions that model real-world situations. The reasoning techniques should be computationally efficient.
- *Usability of modeling formalisms.* Context information models are created by designers of context-aware applications and are also used by the context management systems and context-aware applications to manipulate context information. Therefore, the important features of modeling formalisms are the ease with which designers can translate real-world concepts to the modeling constructs and the ease with which the applications can use and manipulate context information at runtime.
- *Efficient context provisioning.* Efficient access to context information is needed, which can be a difficult requirement to meet in the presence of large models and numerous data objects. To select the relevant objects, attributes for suitable access paths have to be represented in the context modeling. These access paths represent dimensions along which applications often select context information, typically supported by indexes. These dimensions are often referred to as primary context, in contrast to secondary context, which is accessed using the primary context. Commonly used primary CAs are the identity of context objects, location, object type, time, or activity of user. Since the choice of primary CAs is application dependent, given an application domain, a certain set of primary CAs is used to build up efficient access paths (e.g., spatial indexes if location is a primary context).

In Table 12, we position our approach with respect to the above requirements.

- *Heterogeneity and mobility*. Our approach is able to express different types of context information by separately modeling distinct FOCEMs and combining them in HOCEMs.
- *Relationships and dependencies*. Our approach leaves the freedom of creating the best combination of CAs for any application domains. Dependencies among events triggering transitions on FOCEM can be expressed through

Table 12 Evaluation of the CEM modeling approach

Requirements	FOCEM Approach
Heterogeneity and mobility	Fully supported
Relationships and dependencies	Fully supported (remote firings)
Timeliness	Potentially supported but not used
Imperfection	Not supported
Reasoning	Stochastic processes (Markov chains)
Usability of modeling formalism	Inherited by Harel's statecharts
Efficient context provisioning	Not applicable

remote firings [26]. The combination of FOCEMs into HOCEMs is independent from the particular CAs.

- *Timeliness*. FOCEM and HOCEM could support the modeling of context histories through the history mechanism defined in Harel's statecharts [26] and inherited by UML StateMachines [31]. However, due to the memoryless property of Markov chains, i.e., the formalism underlying FOCEM and HOCEM, timeliness is not taken into account in context reasoning.
- *Imperfection*. Not supported. The quality of the CAs cannot be discriminated using FOCEM and HOCEM. However, imperfections are higher for approaches dealing with low-level context data like those monitored by sensors. It is lower for approaches using higher-level context abstractions like situations [13] as we do with FOCEM and HOCEM.
- *Reasoning*. The reasoning capability is based on Markov chains.
- *Usability of modeling formalism*. The usability is inherited from Harel's statecharts [26]. FOCEMs can be easily created by designers with a generic Markov chain editor (e.g., JMT[16] or SHARPE[17]) without the need of learning a new domain-specific language.
- *Efficient context provisioning*. This is a context management system-specific requirement and goes beyond the scope of our approach.

References

1. Afanasov, M., Mottola, L., Ghezzi, C.: Towards context-oriented self-adaptation in resource-constrained cyberphysical systems. In: Proceedings of IEEE 38th International Computer Software and Applications Conference Workshops (COMPSACW), pp. 372–377 (2014)
2. Alti, A., Boukerram, A., Roose, P.: Context-aware quality model-driven approach: a new approach for quality control in pervasive computing environments. In: Proceedings of the 4th European Conference on Software Architecture, ECSA'10, pp. 441–448, Springer, Berlin (2010)
3. Amundsen, S.L., Eliassen, F.: A resource and context model for mobile middleware. Pers Ubiquitous Comput **12**(2), 143–153 (2008)
4. Baier, C., Katoen, J.-P.: Principles of Model Checking. The MIT Press, Cambridge (2008)
5. Balsamo, S., Bernardo, M., Simeoni, M.: Combining stochastic process algebras and queueing networks for software architecture analysis. In: Proceedings of the 3rd International Workshop on Software and Performance, pp. 190–202. ACM (2002)
6. Balsamo, S., Di Marco, A., Inverardi, P., Simeoni, M.: Model-based performance prediction in software development: a survey. IEEE Trans. Softw. Eng. **30**(5), 295–310 (2004)
7. Bencomo, N.: Supporting the modelling and generation of reflective middleware families and applications using dynamic variability. Ph.D. thesis, Computing Department, Lancaster University (2008)
8. Berardinelli, L., Cortellessa, V., Di Marco, A.: Performance modeling and analysis of context-aware mobile software systems. In: Rosenblum, D.S., Taentzer, G. (eds.) Fundamental approaches to software engineering: 13th international conference, FASE 2010, Paphos, Cyprus, vol. LNCS 6013, pp. 353–367. Springer, Berlin (2010)
9. Berardinelli, L., Di Marco, A., Di Paolo, F.: MICE: monitoring and modeling the context evolution. In: SASO workshops, pp. 139–144. IEEE Computer Society (2012)
10. Bernardi, S., Merseguer, J., Petriu, D.C.: A dependability profile within MARTE. Softw. Syst. Model. **10**(3), 313–336 (2011)
11. Bernardo, M., Bravetti, M.: Performance measure sensitive congruences for markovian process algebras. Theor. Comput. Sci. **290**, 117–160 (2003)
12. Bernardo, M., Ciancarini, P., Donatiello, L.: ÆMPA: a process algebraic description language for the performance analysis of software architectures. In: Workshop on software and performance, pp 1–11 (2000)
13. Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A survey of context modelling and reasoning techniques. Pervasive Mob. Comput. **6**(2), 161–180 (2010)
14. Brambilla, M., Cabot, J., Wimmer, M.: Model-Driven Software Engineering in Practice. Morgan & Claypool (2012)
15. Bures, T., Hnetynka, P., Kofron, J., Al Ali, R., Skoda, D.: Statistical approach to architecture modes in smart cyber physical systems, In: WICSA and COMPARCH (2016)
16. Chung, L., do Prado Leite, J.: On non-functional requirements in software engineering. In: Borgida, A., Chaudhri, V., Giorgini, P., Eric, Y. (eds) Conceptual Modeling: Foundations and Applications. Lecture Notes in Computer Science, vol. 5600, pp. 363–379. Springer, Berlin (2009)
17. Cortellessa, V., Mirandola, R.: PRIMA-UML: a performance validation incremental methodology on early UML diagrams. Sci. Comput. Program. **44**(1), 101–129 (2002)
18. Cortellessa, V., Singh, H., Cukic, B.: Early reliability assessment of UML based software models. In: Workshop on Software and Performance, pp. 302–309 (2002)
19. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley, London (2012)
20. Di Marco, A., Mascolo, C.: Performance analysis and prediction of physically mobile systems. In Proceedings of the 6th International Workshop on Software and Performance, WOSP '07, pp. 129–132, New York (2007). ACM
21. DiVA Project. DynamIc VAriability in complex adaptive systems Research Project (2011)
22. Dobson, S., Ye, J.: Using fibrations for situation identification. In: Pervasive 2006 Workshop Proceedings, pp. 645–651. Springer, London (2006)
23. Grassi, V., Mirandola, R., Randazzo, E.: Model-driven assessment of QoS-aware self-adaptation. In: Cheng, B., de Lemos, R., Giese, H., Inverardi, P., Magee, J. (eds.) Software Engineering for Self-Adaptive Systems. Lecture Notes in Computer Science, vol. 5525, pp. 201–222. Springer, Berlin (2009)
24. Grassi, V., Mirandola, R., Sabetta, A.: A model-driven approach to performability analysis of dynamically reconfigurable component-based systems. In: Proceedings of the 6th International Workshop on Software and Performance, WOSP '07, pp. 103–114, ACM, New York (2007)
25. Grassi, V., Mirandola, R., Sabetta, A.: A UML profile to model mobile systems. In: Baar, T., Strohmeier, A., Moreira, A., Mellor, S.J. (eds) UML 2004 The Unified Modeling Language. Modeling Languages and Applications, vol. 3273 of Lecture Notes in Computer Science, pp. 128–142. Springer, Berlin (2004)

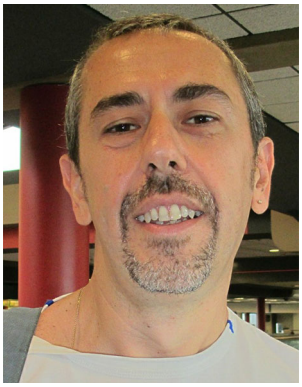
[16] Java Modeling Tools, http://jmt.sourceforge.net/.

[17] SHARPE, http://people.ee.duke.edu/~kst/.

26. Harel, D.: Statecharts: a visual formalism for complex systems. Sci. Comput. Program. **8**(3), 231–274 (1987)
27. Hillston, J.: A Compositional Approach to Performance Modelling. Cambridge University Press, Cambridge (1996)
28. Hirsch, D., Kramer, J., Magee, J., Uchitel, S.: Modes for software architectures. In: Gruhn, V., Oquendo, F. (eds.) Software Architecture. Lecture Notes in Computer Science, vol. 4344, pp. 113–126. Springer, Berlin (2006)
29. Hirschfeld, R., Costanza, P., Nierstrasz, O.: Context-oriented programming. J. Object Technol. **7**(3), 125–151 (2008)
30. Hong, J.-Y., Suh, E.-H., Kim, S.-J.: Context-aware systems: a literature review and classification. Expert Syst. Appl. **36**(4), 8509–8522 (2009)
31. Inc Object Management Group. UML 2.4.1 Superstructure Specification, formal/2011-08-06 (2011)
32. Inc Object Management Group. UML Profile for MARTE, ptc/08-06-09 (2008)
33. Inverardi, P., Mancinelli, F., Nesi, M.: A declarative framework for adaptable applications in heterogeneous environments. In: Proceedings of the 2004 ACM Symposium on Applied Computing, SAC '04, pp. 1177–1183, ACM, New York (2004)
34. Inverardi, P., Tivoli, M.: The future of software: adaptation and dependability. In: De Lucia, A., Ferrucci, F. (eds.) Software Engineering. Lecture Notes in Computer Science, vol. 5413, pp. 1–31. Springer, Berlin (2009)
35. IST-MUSIC Project. Middleware Support for Self-Adaptation in Ubiquitous and Service-Oriented Environments (2013)
36. Kiukkonen, N., Blom, J., Dousse, O., Gatica-Perez, D., Laurila, J.: Towards rich mobile phone datasets: Lausanne data collection campaign. In: Proceedings of the 7th International Conference on Pervasive Services (2010)
37. Kleppe, A.G., Warmer, J., Bast, W.: MDA Explained: The Model Driven Architecture: Practice and Promise. Addison-Wesley Longman Publishing Co., Inc., Boston (2003)
38. Koziolek, H.: Performance evaluation of component-based software systems: a survey. Perform. Eval. **67**(8), 634–658 (2010)
39. Liptchinsky, V., Khazankin, R., Schulte, S., Satzger, B., Truong, H.-L., Dustdar, S.: On modeling context-aware social collaboration processes. Inf. Syst. **43**, 66–82 (2014)
40. Lundesgaard, S.A., Lund, K., Eliassen, F.: Service plans for context- and qos-aware dynamic middleware. In: ICDCS Workshops 2006. 26th IEEE International Conference on Distributed Computing Systems Workshops, 2006., p. 70, July 2006
41. Morin, B., Fleurey, F., Bencomo, N., Jézéquel, J.-M., Solberg, A., Delhen, V., Blair, G.: An aspect-oriented and model-driven approach for managing dynamic variability. In: MODELS'08, vol. 5301 of LNCS, pp. 782–796 (2008)
42. Neuts, M.F.: Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach. Courier Corporation (1981)
43. Rouvoy, R., Barone, P., Ding, Y., Eliassen, F., Hallsteinsen, S.O., Lorenzo, J., Mamelli, A., Scholz, U.: MUSIC: middleware support for self-adaptation in ubiquitous and service-oriented environments. In: Software Engineering for Self-Adaptive Systems, pp. 164–182 (2009)
44. Smith, C.U., Williams, L.G.: Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software. Addison Wesley Longman Publishing Co., Inc., Redwood City (2002)
45. Stewart, W.J.: Introduction to the Numerical Solution of Markov Chains. Princeton University Press, Princeton (1994)
46. Strang, T., Linnhoff-Popien, C.: A context modeling survey. In: Workshop on Advanced Context Modelling, Reasoning, and Management, UbiComp 2004—The Sixth International Conference on Ubiquitous Computing, Nottingham/England (2004)
47. Trivedi, K.S., Sahner, R.A.: SHARPE at the age of twenty two. SIGMETRICS Perform. Eval. Rev. **36**(4), 52–57 (2009)
48. Warrendale International Society of Automotive Engineers. SAE-AS5506: SAE Architecture Analysis and Design Language AADL (2004)

Luca Berardinelli is a postdoctoral researcher in the Distributed Systems Group at Technische Universität Wien (Austria). Prior Technische Universität Wien, he has been a postdoctoral researcher in the computer science and engineering department at University of L'Aquila (Italy) where he held his Ph.D. in Computer Science in April 2011. He has been involved in several national and European research projects in the areas of model-driven engineering, non-functional software validation, and model-based testing, which are his current main research interests. He has published about 25 journal and conference articles on these topics and, currently, he is involved in *U*-test, a H2020 project.

Marco Bernardo received a Ph.D. in computer science in 1999 from the University of Bologna, Italy. Since 2001, he is professor of computer science at the University of Urbino, Italy, where he has chaired for several years the undergraduate program in Applied Informatics and currently is the Rector's Delegate for Technological Innovation. His research interests include: semantics of concurrent programming languages; process algebras, behavioral equivalences, modal logics; labeled transition systems and Petri nets; modeling and verification of concurrent and distributed systems; performance evaluation of computer systems and networks; foundations of software architecture and engineering; automated support for model-based software development. He has co-authored a book entitled "A Process Algebraic Approach to Software Architecture Design" as well as more than twenty papers published on international scientific journals and more than seventy papers published on the refereed proceedings of international conferences.



Vittorio Cortellessa is an associate professor in the computer science and engineering department at University of L'Aquila (Italy). Prior joining University of L'Aquila, he has been Post-doc Fellow at the European Space Agency (Roma, Italy), Research Associate at the Computer Science Department of University of Rome Tor Vergata, and Research Assistant Professor at the Lane Department of Computer Science and Electrical Engineering of West Virginia University (Morgantown, WV). He has been involved in several research projects in the areas of performance and reliability analysis of software/hardware systems, model-driven engineering and non-functional software validation, which are his current main research interests. He has served as reviewer for many journals on his research topics. He has published about 100 journal and conference articles on these topics. He is member of editorial boards of journals, he has served and is currently serving in the program committees of conferences in his research areas.

Antinisca Di Marco is associate professor at the University of L'Aquila where she held her Ph.D. in Computer Science in June 2005. Since May 2014, she is a Member of the Board of Directors for SMARTLY s.r.l., a SPIN OFF of the University of L'Aquila and a Member of the Executive Board of Off Site Art, an association established in L'Aquila that promotes culture initiative. Previously she worked as assistant professor at the same University and as Research Fellow at the University College London, UK. Her main research interests include (early) verification and validation of QoS, performance modeling, QoS analysis of autonomic services and context-aware mobile software systems, bio-inspired adaptation mechanisms, eHealth, and Bioinformatics. She published more than 50 journals and conference papers on such topics. She has served as program committee member for several international conferences and workshops, and as reviewer for many journals on her research topics. She has been member and coordinator of several national and international research projects and, currently, she is involved in the iCARE, a H2020 ERC-POC project.