Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

RESEARCH ARTICLE

# A procedural and flexible approach for specification, modelling, definition and analysis for self-adaptive systems

Patrícia Araújo-Oliveira[1,1] | Francisco Durán[1] | Ernesto Pimentel[1]

[1]ITIS Software, University of Málaga, Málaga, Spain
[2]Department of Exact Science and Technology, Federal University of Amapá, Amapá, Brazil

**Correspondence**
Patrícia Araújo-Oliveira, University of Málaga. Málaga, Spain.
Email: patricia.araoli@gmail.com

**Summary**

An adaptive system can modify its settings at runtime as a response to changes in its operational environment. To analyse this kind of systems at design time is a difficult task since it requires considering the system together with the adaptation operations, and taking into account how such adaptations act on the system. In order to use simulation-based techniques for the analysis of such systems, we not only need precise executable models of the systems to be analysed, but also to capture the semantics of their adaptation mechanisms. Given the wide range and flexibility of adaptation operations, we need ways to allow the definition of new operations. We present a flexible approach for the definition and simulation-based analysis of adaptive component-based systems. Our approach combines an extension of the Palladio component model in e-Motions, a model of the adaptation mechanisms, and elastic requirements specification using the SYBL language. From the model of the system, its adaptation mechanisms, and its requirements, an executable Maude specification is automatically generated for simulation. The application of the approach is illustrated on a use case that comprises some components and adaptations rules. The example is then analysed using simulations. It is also shown that it is indeed possible to define additional metrics, specify adaptation requirements and rules which conduct simulations of the models in a more flexible way, and that the results of the simulation performed from these definitions can be used to carry on a valuable predictive performance analysis.

**KEYWORDS:**
adaptation specification, modelling, self-adaptive models, self-adaptive behaviour, design-time analysis

## Contents

# 1 | INTRODUCTION

Software architecture has become an essential element in building consistent systems as the size and complexity of software systems have increased. The necessity to take into account not only the overall structure of the system itself but also the allocation, functionality and communication of the software components became a challenge for building software architecture projects. These definitions must establish the system structure consistently for implementations and are directly related to quality attributes, such as reliability and performance (non-functional requirements). Providing services efficiently without delays or failures is crucial for any software system, as it directly impacts the user's experience.

Koziolek[?] and Balsamo et al.[?] indicate that the predictive analysis of these non-functional requirements in design time might mitigate development costs and failure risks. Indeed, discovering at late stages of the development process that a software system does not meet certain non-functional requirements can be very harmful. The late identification of these issues may require changes that can aggravate the situation, both in terms of costs and risks.

The management of these characteristics in a self-adaptive system is a challenge that, in addition to complexity, has dynamism as its main feature. It is necessary to consider possible modifications of its initial configurations in response to changes in the context, which can compromise its functioning due to some situations, such as, for example, inconsistent context, poor performance and failures to adapt. This makes the construction of these systems and decision-making at design time difficult tasks, since the initial model can take different forms and configurations during the execution process.

## 1.1 | Motivation and Challenges

Sousa et al.[?] show that the studies related to the performance of self-adaptive systems has been amongst the most popupar ones within the academic community in the last years. They present a series of challenges for quality evaluation of self-adaptive systems regarding architecture, complexity, reference values and tools. We consider that possible paths for overcoming these issues rely on the effort to build solutions at design time.

In relation to system architecture, the challenges of not being able to verify all the adaptation possibilities[??] and the difficulty of evaluating different architectures[?] might be the main issues. Several proposals have tried to address it by statically studying, at design time, the performance of different adaptation alternatives and architectures. These results could then assist in decision-making at runtime, since the system could be equipped with predefined adaptation paths and could be prepared to choose the most appropriate adaptations for a given situation. These techniques would also facilitate the construction of initial reference values to assist in the interpretation of results, which is pointed out as another challenge.[???]

However, these possible solution paths lead us to another challenge, pointed out by the same authors: *the complexity of a self-adaptive system must be considered in simulations and there is a difficulty in reproducing these conditions as they exist in real-world application scenarios*.[??] Thus, research paths that strive to advance in the areas of performance analysis of self-adaptive systems at design time are fundamental in an attempt to overcome problems still present in the area. Projects in this direction must consider the typical complexity of these systems and the proposal of tools to automate and assist in the tasks of tests and quality assessment (also pointed out as a challenge by Franco et al.[?] and Bezerra et al.[?]).

Raibulet et al. has recently presented an overview of some existing approaches for the evaluation of self-adaptive systems.[?] According to their analysis on the published works, when quality attributes (such as performance and reliability) are considered, these attributes are evaluated at runtime in 95% of the cases, and at design time in 5% of them. The authors also pointed out that none of these approaches associated these evaluations to a tool, which may hamper the evaluative part.

Indeed, *few studies and tools are found in the literature that address the question of evaluating the performance of self-adaptive systems at design time*. As for tools, Palladio[?] is one of the currently most successful predictive analysis frameworks, and is widely used both in industry and academia. Although the Palladio Simulator SimuCom[?] can be used to predict Quality of Service (QoS) properties (performance and reliability) from software architecture models, it is limited to the analysis of static systems. SimuLizar, proposed by Becker et al.,[?] was intended to extend Palladio for the performance analysis of self-adaptive systems at design time. Even though the approach is quite effective, the behaviour of the system and adaptations are encapsulated in the platform, which hinders the insertion or modification of new adaptation strategies or different application domains. It is indeed an important drawback, considering the constant evolution and change in the application of self-adaptive systems.

## 1.2 | Research Questions and Hypotheses

This paper aims to answer the following research question: *how to incorporate possible changes in the models of self-adaptive systems to the analysis at design-time?* In order to take into account system changes, attention needs to be paid to uncertainties in the behaviour of systems, which typically come with unpredictability. The uncertainties, for example, could be probabilistically modelled using stochastic processes. However, *how to model and measure the unpredictability of self-adaptive models?* We understand that an analysis of self-adaptive models that considers unpredictability must be the result of the analysis of a set of values obtained from the understanding of the possible changes in the model itself.

In order to answer the previous questions, *we propose the use of a procedural approach relying on the use of a set of tools to enable flexibility in the specification, system modelling, behavioural definition, and performance analysis*. Specifically, we propose using Palladio to model the system to be analysed, e-Motions[?] to model metamodels and behaviours, and the SYBL language[?] to specify non-functional requirements and adaptation strategies. Given a specification of both the metamodel and behaviour of Palladio in e-Motions, the entire adaptive system, together with its SYBL execution requirements, is then transformed into a Maude executable specification. This specification is finally used to carry on the simulations using the Maude tools.

Our hypothesis is that, with this approach, *we can obtain a refinement of the results of the analysis of the model through the viabilisation of alternative decisions* based on the verification of changes in the initial settings in response to context changes. Thus, our research path is the search for a flexible approach that *incorporates the changing nature of self-adaptive systems into the design-time analysis*, which can increase the predictability of models of this type of systems. This changing nature consists in the possibility of incorporating the constant changes in the model into its specification, modelling and behavioural definitions, as well as into the performance evaluation, which needs to be continuous and analytical.

## 1.3 | Main Contributions

In order to achieve our objectives, we need to advance the implementation of the previous proposal and develop mechanisms for specifying adaptation strategies and the definition of QoS metrics that facilitate their modification. We choose the SYBL language to control and specify monitoring metrics, restrictions and adaptation strategies, in addition to developing a metamodel and rules of behaviour — using graph transformation with graphical representation of e-Motions — to guarantee the construction and modification of mechanisms for the evaluation of self-adaptive systems at design time, *enabling the identification of the factors that contributed to the results achieved, and, from these, to define actions for the adjustment* in the pursuit of a better performance.

This work builds on our previous work and the work of other authors. Specifically, its main contributions include *the provision of support for the simulation of the application and the environment with adaptability control, parameter adjustment, or reformulation of the established mechanisms through the verification of the impact of the adaptations in the system model*. The use of a graphical notation, namely graph transformation, to model the behaviour of the application, its adaptations, and the evaluation mechanisms, facilitates its modification and understanding, and makes the approach more flexible.

The results of the simulations using our procedural and flexible approach showed us that it is possible to incorporate changes in models of adaptive systems and that our approach is useful to model and measure the unpredictability of these systems, since the composition/integration of different services requires a detailed analysis of the adaptation choices. It is necessary to pay attention to simpler service models to then refine the model and improve its interpretation.

The paper is structured aiming to present, step by step, the specification, modelling, definition and analysis of self-adaptive systems within our proposal. Section **??** provides some background on the Palladio and e-Motions frameworks, and also presents a quick overview of our previous work and the advances to be presented in this article with the motivating example. Section **??** presents SYBL and shows how to use it to specify adaptation strategies using it. Section **??** illustrates the use of Palladio for application modelling and presents our running example. Section **??** explain how to use e-Motions to define the structure and behaviour of languages. Specifically, we introduce the metamodel and behaviour of Palladio using e-Motions, as well as the rules of adaptive behavior and QoS metrics. Section **??** presents a guideline to predictive analysis using our approach. Section **??** presents some related work. We wrap up with some conclusions and future work in Section **??**.

## 2 | OVERVIEW OF THE APPROACH

In this section, we present an overview of our approach and the tools it relies on. Firstly, we present Palladio and e-Motions. Then, we show the advances in the architecture of the approach as well as the process model that aggregates the previous work and the new contributions.

### 2.1 | The Palladio and e-Motions Tools

This work builds on two Model-Driven Engineering (MDE) frameworks: Palladio and e-Motions. We use Palladio[?] for the modelling of the system's architecture and its components, and e-Motions[?] for the specification of the adaptive operations and the control of non-functional properties of the system's components. We build on the e-Motions definition of Palladio by Moreno-Delgado et al.,[?] which was modified to integrate the new capabilities.

Palladio is a tool that provides facilities for the analysis of the performance of systems. The metamodel of Palladio — its language structural description — is provided by the Palladio Component Model (PCM).[?] In Palladio, the semantics of the models, and the non-functional properties to be analysed, are encapsulated in respective transformations to different languages and formalisms in which the analysis is carried out. Moreno-Delgado et al. developed a complete definition of the language by attaching an operational description of Palladio models through visual graph transformation rules.

SimuLizar[?] is a tool based on Palladio for the performance analysis of adaptive systems. The tool supports the modelling of systems with adaptation requirements and their analysis. However, it does not allow the insertion of new quality metrics nor the definition of adaptation mechanisms beyond the predefined ones. Furthermore, the model cannot be modified during the analysis. From the initial model, the changes applied to it for adaptation occur from the resetting of parameters only, and it is not possible, for example, to add a new component or even a new container and its network connections. This makes it difficult to advance on the way to meet the needs, for example, of a smart-city scenario where Internet of Things (IoT) devices act on cloud, edge or fog environments, and new components and devices are inserted at any time.

Considering the limitations of Palladio, the e-Motions system[?] was another tool that we used in the formulation of a more flexible approach for the performance analysis of self-adaptive systems. It is a graphical framework that supports the specification, simulation, and formal analysis of systems. It provides a way to graphically specify the dynamic behaviour of Domain-Specific Languages (DSLs) using their concrete syntax, making this task very intuitive. The abstract syntax of a DSL is specified as an Ecore metamodel, which defines all relevant concepts and their relations in the language. Its concrete syntax is given by a Graphical Concrete Syntax (GCS) model, which attaches an image to each language concept. Then, its behaviour is specified with (graphical) in-place model transformations.

The e-Motions language provides a model of time, supporting features like duration, periodicity, etc., and mechanisms to state action properties. From a DSL definition, the e-Motions tool generates an executable Maude[?] specification which can be used for simulation and analysis.[?] For instance, we can perform reachability analysis, model checking, and statistical model checking of the DSLs defined using e-Motions.[??]

The in-place model transformations used to specify the behaviour of systems are defined by graph transformation rules, each of which represents a possible *action* of the system. These rules are of the form

$$[\text{NAC}]^* \times \text{LHS} \rightarrow \text{RHS}$$

where LHS (left-hand side), NAC (negative application conditions) and RHS (right-hand side) are model patterns that represent certain (sub-)states of the system. The LHS and NAC patterns express the conditions for the rule to be applied (the definition of NAC patterns is optional), whereas the RHS represents the effect of the corresponding action if its conditions are satisfied. Thus, the action described in RHS can be applied. I.e., a rule can be triggered if a match of the LHS is found in the model and none of its NAC patterns occurs. An LHS may also have positive conditions, which are expressed, as any expression in the rules, using the Object Constraint Language (OCL). If several matches are found, one of them is non-deterministically chosen and applied, giving place to a new model where the matching objects are substituted by the corresponding instantiation of its RHS pattern. The transformation of the model proceeds by non-deterministically applying the rules on sub-models, until no further transformation rule is applicable.

## 2.2 | Previous Work and Advances

Moreno-Delgado et al. proposed an approach[?] that allowed the analysis of Palladio models of systems. Their work used the metamodel of Palladio, the PCM, and its operational semantics expressed in terms of graph-transformation rules. An automatic transformation from e-Motions to Maude was used to obtain executable Maude specifications from Palladio models, which was then used to carry on a simulation-based analysis of its performance. This implementation focus on representing a flexible approach to Palladio using graph-transformation rules. Therefore, like Palladio, the approach allows only analysis of static systems, with no support for dynamic model change.

To enable performance analysis capabilities for adaptive dynamic systems, we used the e-Motions implementation of Palladio and the graph-transformation approach to also model the adaptive (and elastic) behaviour of systems. The partial implementation made by Moreno-Delgado et al. was extended with additional rules to also support communications and other system features not contemplated in their work, which were strategic for our goals.

The first results of our approach were presented 2017,[?] with basic support for scale up and scale down adaptations (vertical adaptations) that increase or decrease CPU capability, aiming to check whether it was possible to add adaptation rules to the e-Motions implementation of Palladio. Then, in 2018,[?] we improved our approach with horizontal adaptations: the scale in and scale out adaptations of servers along with scale up and scale down mechanisms. In both works, the adaptations were directly specified in the behaviour rules, and executed in simulation time in response to the violations of constraints. This solution allowed us to show the possibilities of the approach, but the adaptation operations were hardwired on the rules specifying the adaptations themselves, hindering their specification and lacking the required flexibility.

In this current work, we have opted for a separated specification of the adaptation operations, and of the non-functional requirements and the way such operations must be executed. Instead of developing our own ad-hoc language, we use SYBL,[?] a specification language for multi-level elasticity control. SYBL covers different elasticity constraints, strategies, and monitoring directives, and supports a wide range of flexible ways for controlling the elasticity of applications. By specifying appropriate SYBL annotations, we can then model the mechanisms in some providers, and experiment with any other combination of adaptation mechanisms available, thus increasing the scope of the performance analysis.

We also operated the integration of the e-Motions Observers with Monitors and the development of MAPE-K control rules along with an adaptation control. In this way, the work presented in this paper substantially expands our previous efforts.

## 2.3 | Artefacts and Processes of the Approach

The diagram in Figure **??** depicts the main elements in our proposal. The upper left square represents Palladio's dual role. The first role is in building the system's models to be analysed ($M_{app}$). Models conforming to the PCM are composed of four different sub-models, which correspond to the respective views of systems (more details in Section **??**). The second role is the representation of its abstract syntax (the PCM).

The right side represents the role of the e-Motions language and tool. Palladio is specified in e-Motions both through its syntax — an extended PCM denoted $PCM^*$ — and through its behaviour — denoted $Beh_{Palladio}$. The static systems defined in Palladio (models $M_{app}$ conforming to the PCM) can be loaded and analysed in e-Motions using the DSL $PCM^* + Beh_{Palladio}$. To deal with adaptive systems, the Palladio behaviour was extended, including adaptation mechanisms specified as additional e-Motions rules ($Beh_{Adaptation}$).

This allows us to use available Palladio models, or use the Palladio tools to design them ourselves.

The control of the execution of systems thus described is guided by SYBL Annotations (represented in the lower left corner). SYBL Annotations give place to corresponding specifications $Ctrl_{SYBL}$. The combined $M_{app} + Ctrl_{SYBL}$ conforms to the DSL metamodel $PCM^* + (Beh_{Palladio} + Beh_{Adaptation})$, which are then transformed into Maude using the e-Motions tools. The generated Maude code can then be used for simulation. Finally, the results obtained from the simulations are used for performance analysis.

## 2.4 | Operational Flow of the Approach

The architecture presented in the previous section illustrates how the different tools used in our approach are integrated. In this section, we show its operational flow. Figure **??** shows a process model (using basic standard BPMN notation[?]), in which we can see the role of each tool and the different activities of the process: Specification, Modelling, Definition and Analysis.

In the Specification phase, the non-functional requirements and adaptation strategies are specified using the SYBL language (more details in Section **??**). In the Modelling phase, the initial model of the system to be analysed is modelled in the Palladio

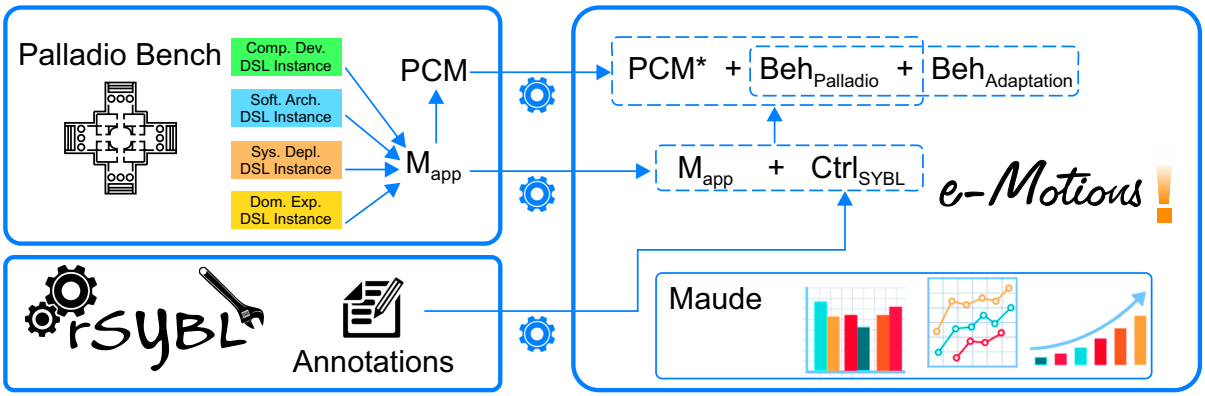



**FIGURE 1** Artefacts and processes of the approach

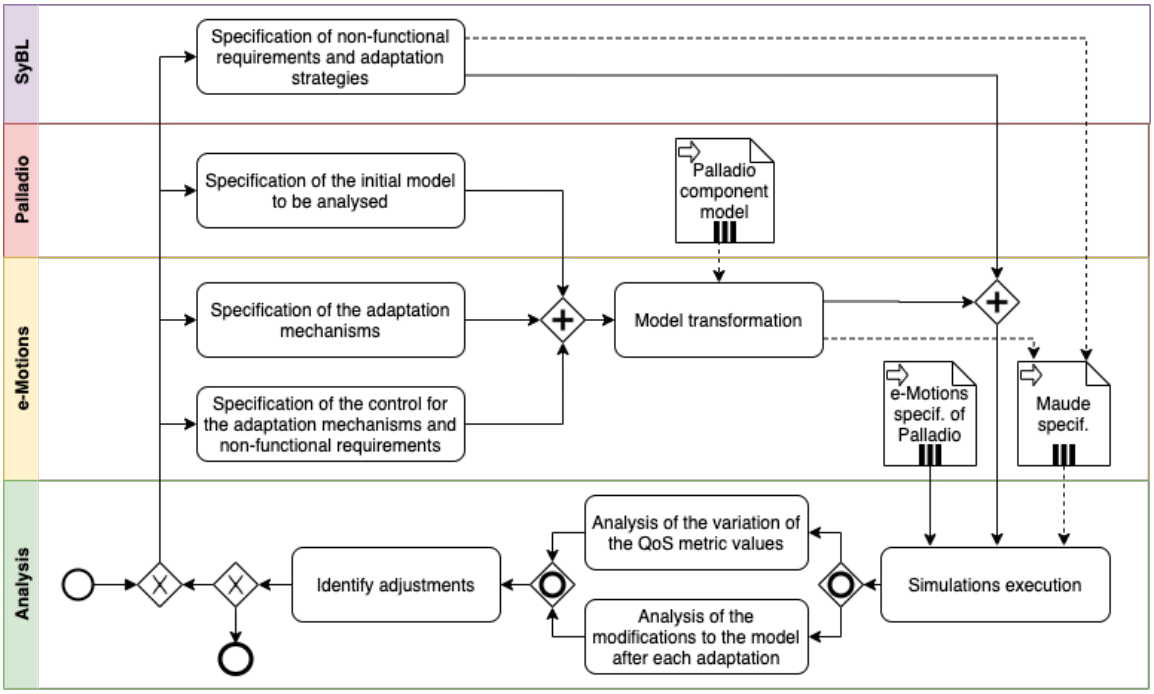



**FIGURE 2** Procedural view of the approach

tool, in which we obtain the different views of the application (more details in Section **??**). This model can be defined according to the specifics of each application to be analysed.

In the Behaviour Definition phase, the Maude code is generated. It is obtained by a model-transformation process provided by the e-Motions tool. The Maude specification obtained can then be used to carry out the simulation in the Analysis phase. As explained in the previous section, the e-Motions transformation takes as input: (1) the initial model, defined in Palladio in the Specification phase; (2) the Palladio Component Model and its e-Motions behavioural definition (more details in Section **??**); (2) the adaptation mechanisms (more details in the section **??**); and (3) the adaptation and non-functional requirements control (see Section **??**).

In the Analysis phase, the Maude system is used to execute the simulations of the Maude codes generated in the previous phase. During and at the end of the simulations, it is possible to view the simulation results, which contain the data related to the simulated system. This data shows the values of the variation of the QoS metrics monitored during the simulation as well as, if applicable, the modifications to the model that occurred after the performed adaptations. This data allows the analysis and, later, the adjustments in the initial model or in the parameters, as needed.

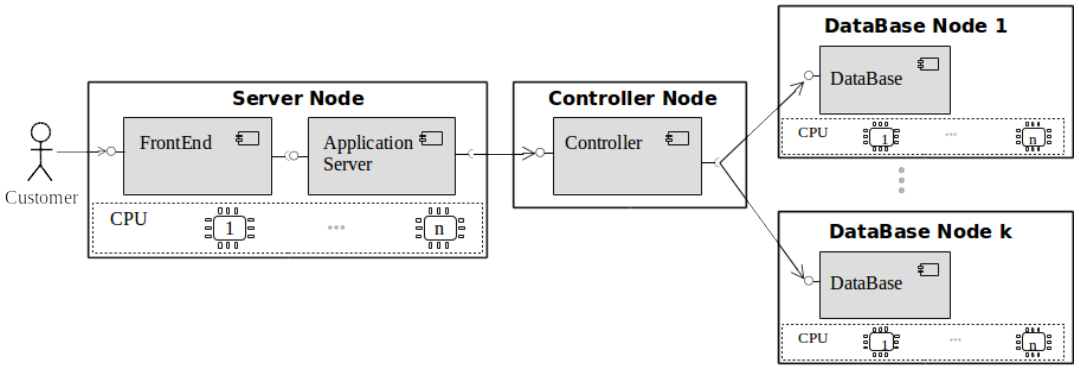



**FIGURE 3** Structure of the example

## 3 | SPECIFICATION PHASE: SYBL ANNOTATIONS

We use SYBL[?] for the specification of monitoring, constraints and adaptation strategies. SYBL is a popular language designed specifically for these purposes, which make it a good choice. The separated specification of adaptation requirements allows us to simplify both the modelling of systems and its adaptation mechanisms, at the time we improve the flexibility and maintainability of the adaptation management. Furthermore, their isolated development allows us to reuse them when possible. In this section we give a brief introduction to SYBL, discuss our motivating example, and show how to proceed in the writing of the specification of the adaptation strategies, monitoring and constraints using SYBL.

### 3.1 | Motivating Example

With the purpose of illustrating our approach, we use a case study, called "Shop Data System", which we present in this section. The example is inspired on the one presented by Becker et al.[?] It models a shopping site in which the customer enters a web page, and, when purchasing a product, the system processes and forwards the request to the database nodes to complete the process. Despite its simplicity, which may help the reader to understand all its details, it includes elements commonly present in a web system: web servers, application servers, load balancers and databases.

Given the above description, Figure **??** shows the structure of the example. The following components will be designed: Front End represents the Web Server with the HTTP service; Application Server represents the process request service; Load Balancer is responsible for the access control to the database, which we call DataBase Controller; and the DataBase component, with the data process service. The structure in Figure **??** not only shows these components, but it also shows the allocation of these components. It shows that each of the components is to be allocated in its own node, but the FrontEnd and Application Server, which will be allocated together in the Server node. The diagram also includes information on resources. As we will see below, the Server node only allows vertical scaling, that is, we can only increase or decrease the processing capacity of its CPU resource. However, the Controller and the DataBase nodes are allocated to cloud providers that allow to scale not only vertically, but also horizontally (i.e., to add nodes and allocate components on them).

To simplify the abstraction, we consider that all user requests will need the same demand for resources. However, they will be different for the Application Server and DataBase components. Of course the workload will change along the simulations. As an initial model, we will have three nodes: Server, Controller and DataBase Node 1.

The following requirements (R1-R4) will be considered:

R1: The response time for a user request should not be, on average, greater than 0.5 time units (t.u.).
R2: The percentage of resource usage for nodes should not be greater than 65%.
R3: If R1 is not met, that is, if the average response time is greater than 0.5 t.u., the system must restore an average response time of 0.5 t.u. or less as soon as possible.
R4: If R2 is not met, that is, if the percentage of resource usage of the nodes is greater than 65%, the system should re-adjust to decrease the workload on the nodes as soon as possible.

Listing 1: Component-level SYBL annotation for ApplicationServer Component

```
1  @SYBL_ComponentContext(
2    ComponentID = Component3;
3    ComponentName = ApplicationServer;
4    constraints = "Co1: CONSTRAINT cpuUsage < 65;
5                   Co2: CONSTRAINT cpuUsage > 30;
6                   Priority(Co1) = 2,
7                   Priority(Co2) = 1",
8    monitoring = "Mo1: MONITORING cpuUsage = ObResourceUsage"
9    strategies = "St1: STRATEGY WHEN Violated(Co1): ScaleUp;
10                  St2: STRATEGY WHEN Violated(Co2): ScaleDown")
```

## 3.2 | SYBL Annotations

SYBL[?] is a language for controlling elastic capabilities of cloud applications at runtime, where elasticity is not only referred to resource scalability, but also to cost and QoS properties. In this way, SYBL is capable of expressing conditions combining these three dimensions. Another feature of SYBL is its ability to enable different kinds of users (application developers, software providers, IaaS/PaaS end-users or cloud providers) to specify the elastic features of an application. The language was designed to be API independent, and it provides a control service that can use different monitoring tools and cloud APIs, regardless of the technology used. Furthermore, SYBL was designed to be extensible, that is, it facilitates the involvement of new concepts and allows to link metrics or syntactic entities of the language to external elements that can correspond to measurable variables or units in the system being modelled.

The SYBL expressiveness concerns not only to the multi-dimensional elasticity (resources, cost and quality) but also to the multi-level description possibilities. Thus, it is possible to specify different elasticity constraints and strategies at the application, component and programming levels, by means of the so-called *directives*. An annotation is a set of different directives, and represents the requirement specifications for an application and each of their components. For each annotation, we need to indicate: the specification level (application, component or programming), and a label for defining the monitoring metrics, for describing the constraints to be taken care of, and for establishing the strategies to apply when necessary. In addition, other information could also be provided (identifier, component name, priorities of constraint and strategies, etc.).

Listings **??** and **??** show samples of SYBL annotations with resource usage and response time quality metrics, respectively. Both are component-level annotations, which specify a ComponentID, its ComponentName, and the description of constraints, monitoring and strategies. The constraints section of the annotations includes the specification of conditions and their priorities. Each condition indicates the expected threshold for certain monitoring metrics — conditions on the CPU usage (cpuUsage) in Listing **??**, lines 5-6, and on the response time (rt) in Listing **??**, lines 5-6. Metrics referring to resources usage are given as percentages, whereas all of them alluding to time are considered as time units. The strategies section provides the actions to be performed when constraints are violated. The St1 strategy in Listing **??**, line 9, will produce a scaling-up operation when the CPU usage is over 65% (condition Co1 is violated), while the St2 strategy (line 10) will produce a scaling-down operation when the CPU usage is under 30% (condition Co2 is violated). Similarly, the strategy St3 in Listing **??** will produce a scaling-out operation when the response time of the service offered by the DataBase component is greater than 0.5 (condition Co3 is violated) and the St4 strategy will produce a scaling-in operation when the response time is under 0.2 (condition Co4 is violated).

Note that monitored metrics (rt and cpuUsage in the example), observers (ObResourceUsage and ObResponseTime) and response operations (ScaleUp, ScaleDown, ScaleIn and ScaleOut) are references to elements in the system model and the adaptation operations. We will see how these references appear in the other components in Section **??**.

# 4 | MODELLING PHASE: PALLADIO VIEWS

We build the system models using Palladio. In this section we present each model and contextualise it with Palladio's conceptual definitions.

Palladio assumes a Component-Based Software Engineering (CBSE) development process, in which component developers specify and implement parametric descriptions of components and their behaviour. These descriptions are organised in four different views:[?] component developers provide *component specifications*; software architects provide *assembly models*; system developers provide *allocation models*; and business domain experts provide *usage models*. To illustrate these Palladio views, we

Listing 2: Component-level SYBL annotation for DataBase Component

```
@SYBL_ComponentContext(
  ComponentID = Component4;
  ComponentName = DataBase;
  constraints = "Co3: CONSTRAINT rt < 0.5;
                 Co4: CONSTRAINT rt > 0.2;
                 Priority(Co3) = 2,
                 Priority(Co4) = 1",
  monitoring = "Mo2: MONITORING rt = ObResponseTime",
  strategies = "St3: STRATEGY WHEN Violated(Co3): ScaleOut;
                St4: STRATEGY WHEN Violated(Co4): ScaleIn")
```

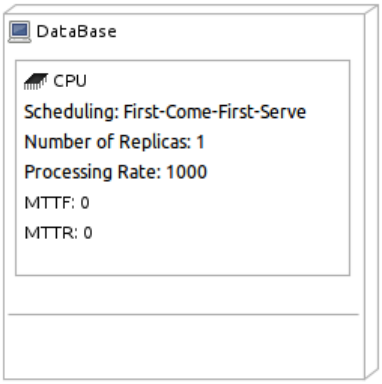



**FIGURE 4** Palladio modelled container

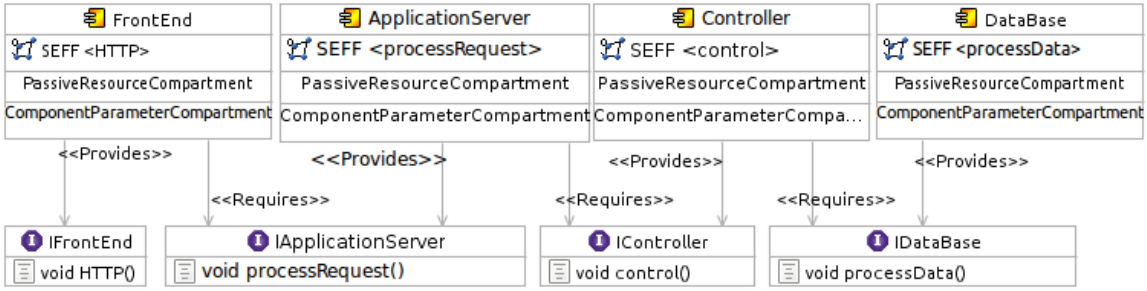



**FIGURE 5** Component Model

present the modelling of the scenario presented in Section **??** with four components: a front end that has the role of receiving the workload; an application server which processes all arriving requests; a database controller that has the role of load balancer and sends the requests to the database component according to the specification; and, finally, the database component processes the requested data.

Figures **??**-**??** are actual snapshots of the specification of the system defined using the Palladio Bench. In Section **??**, we use our adaptation rules to increase and decrease resources as well as to add and remove nodes when necessary, using the SYBL annotations defined in the previous section, from this initial model.

Figure **??** shows a node example modelled in Palladio which is a container with processing resource specification of the CPU type. The adaptations used in this paper will act directly on the containers, whether by adding a new container or by changing the number of replicas of the resource.

In Palladio, component models describe the components and interfaces in the system and the relations between them. Figure **??** shows the component repository for our example. It depicts four components and their corresponding interfaces: FrontEnd implements IFrontEnd, ApplicationServer implements IApplicationServer, Controller implements IController and DataBase

implements IDataBase. There are three Requires relations: (1) from the FrontEnd component to the IApplicationServer interface, that offers the processRequest() operation; (2) from the ApplicationServer component to the IController interface, that offers the control() operation; and (3) from the Controller component to the IDataBase interface, that offers the processData() operation.

Components' services are described by service effect specifications (SEFF), which abstractly model the externally visible behaviour of a service with resource demands and calls to required services. Figure **??** shows the SEFF of the HTTP() service, which models the behaviour of the FrontEnd component that contains an external call action to the ApplicationServer component. Figure **??** shows the SEFF of the processRequest() service, which models the behaviour of the ApplicationServer component, processes an internal action that consumes 200 units of CPU (CPU cycles) and carries out an external call action to the Controller component. Figure **??** shows the SEFF of the control() operation, which models the control flow in the Controller component as a probabilistic branching. Figure **??** shows the SEFF of the processData() service, which models the behaviour of the DataBase component — such processing consists in an internal action that consumes 300 units of CPU (CPU cycles).

We will see in the coming sections how the dynamic extensions are in charge of increasing/decreasing the resources available or adding/removing nodes as needed. Since there is only one branch in this initial Palladio definition, the probabilistic branching of the Controller component starts at 100%. This branch has an external call action to the DataBase node of the model. As new nodes are added to the architecture, new branches will be added to this action, thus modelling the distribution of works between the existing servers handled by the controller. Note that this type of operations is not possible neither in Palladio nor in its SimuLxizar extension for adaptive systems.

Software architects assemble components from the repository to build applications, represented by assembly models in Palladio. Figure **??** shows how the services of components are organised. The biggest square surrounding the boxes represents the entire environment. For each provides relation in the repository model (Figure **??**), an assigned role is created for the container with such component.

Allocation models are provided by system deployers, who model the resource environment and the allocation of components from the assembly model to different resources of the environment. Figure **??** shows the initial allocation model for our example, where we can see how each of the components is allocated in the nodes. The FrontEnd and the ApplicationServer components are allocated in the same node, whereas the Controller and DataBase components are allocated in different nodes.

Finally, usage models are provided by domain experts, who specify a system's usage in terms of workload, user behaviour, and parameters. Given the usage model definition in Figure **??**, in our example, tasks will arrive following an exponential probability distribution with parameter 5.0 time units (Exp(5.0)).

The use of these Palladio models in our approach was facilitated by the work by Moreno-Delgado et al.,[?] which allowed the transformation of the Palladio system models into Maude code, using the e-Motions tool. In this work, we did not make changes regarding the system modelling and model transformations.

## 5 | BEHAVIOUR DEFINITION WITH E-MOTIONS

The main characteristic of our approach is the use of explicit specifications of the behaviour of Palladio and of the adaptation operations available. Since these specifications are modifiable, users have absolute control on the models and the operations available. In this section we present the e-Motions specification of Palladio and the rules for adaptation control and the specification of non-functional requirements. For more information on the e-Motions definition of the Palladio DSL and its extension, please, see https://www.scenic.uma.es/GTPAAS/.

### 5.1 | The Palladio Component Model in e-Motions

Palladio is a DSL, and its specification using the e-Motions system[?] was first proposed by Moreno-Delgado et al. in 2014.[?] This initial specification was then extended and improved as part of our project in successive years. As for any DSL, the e-Motions definition of Palladio includes its abstract syntax (the PCM), its concrete syntax, and its behaviour. Its concrete syntax is provided in e-Motions by a GCS model in which each concept in the abstract syntax being defined is linked to an image. Its behaviour is defined by graph transformation rules, thus becoming explicit at a very high level of abstraction. The fact that the same graphical


**(a)** FrontEnd **(b)** AppServer

**(c)** DataBase Controller **(d)** Data Base

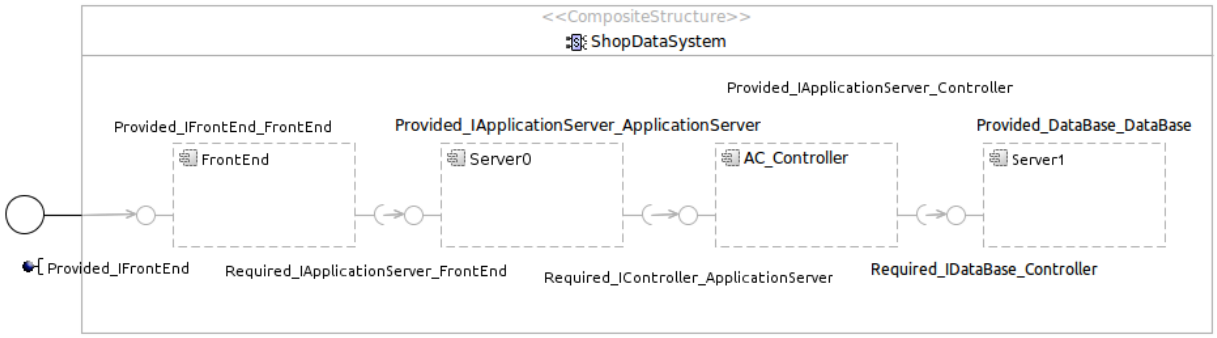
**FIGURE 6** Components SEFFs

representation used in the Palladio Bench are used to graphically represent elements in Palladio models in e-Motions makes these rules very intuitive to users familiar with Palladio.[1]

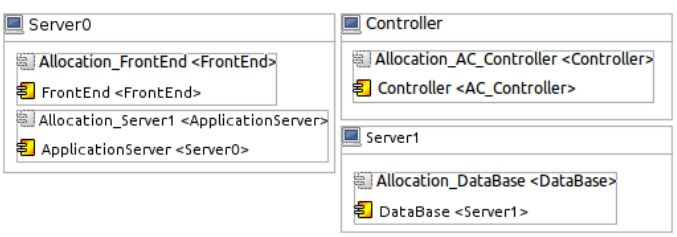
The operational semantics of Palladio, i.e., its behaviour, is given as a token-based execution model, where each work that enters the system is modelled as a token that moves around the different services of the system. Then, inside each service description, tokens move around the different tasks (start, stop, branch, loop, etc.) in its SEFF descriptions. Each of the actions that may occur in the system can be then specified by e-Motions transformation rules.

Let us illustrate the e-Motions rules by showing two of the rules that define Palladio's behaviour. The first one, in Figure **??**, is the OpenWorkloadSpec rule, which models the action of initiating a new execution. In it, we see how a usage scenario (usSc) is linked to a scenario behaviour (scBeh) that defines the sequence of actions to perform and a specification of the workload description. In this case, the usage scenario is described by an open workload (ow), which has a timer associated. The rule models the arrival of a new work into the system. Specifically, when the timer comes to zero, a new token is associated to the start action of the scenario behaviour. The timer is set to the amount specified by the corresponding stochastic expression —

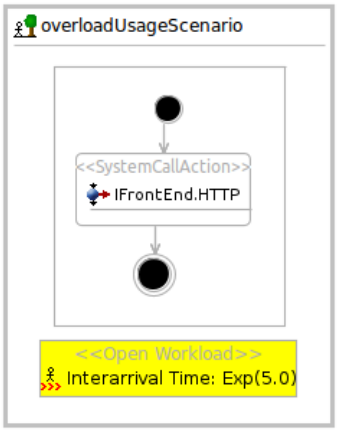
[1]Figures **??**, **??**, **??** and **??** show snapshots of actual e-Motions rules in the specification of Palladio. They include an attached 'legend' section to help the reader not familiar with the PCM.




**(a)** Assembly Model




**(b)** Allocation Model




**(c)** Usage Model

**FIGURE 7** Assembly, Allocation and Usage Models

e.g., Exp(5.0) in our running example. Note the use of the owRate variable, which temporarily holds the result of the evaluation of the given expression.

Figure **??** shows the e-Motions rule that specifies the execution of an InternalAction, like the ones used in Figures **??** and **??**. This rule represents a generic execution of an internal activity by a component service, possibly using a resource, like a HDD or CPU, following an FCFS (First Come First Served) strategy. In Palladio, these executions present a high level of abstraction, and the resource demands are described as stochastic expressions. In the e-Motions rule, the LHS indicates that if there is an internal action (IA) in the system linked to a token (t) with the completed attribute with value false, the RHS will execute in time rTime, calculated using the expression of the header of the rule (PRD / PRS), i.e., the duration of this action depends on the corresponding Palladio elements, specifically on the PRD (Parametric Resource Demand) and on the PRS (Processing Resource Specification). For instance, in our example the IA was specified with PRD of 300 units of CPU (Figure **??**) — the resource type is defined in the object PRT (Processing Resource Type). The container in which the component that specifies this IA was allocated (Figure **??**) presents the processing rate (PRS) of 1000 units of CPU per time unit. Thus, each work will take 0.3 time units to be executed. Tokens are served following an FCFS strategy by using a queue associated to each resource type. Only the first PRS.numberOfReplicas tokens in the queue PRS.queue get to be executed. Once an internal action is executed, its token is removed from the queue (PRS.queue→excluding(t)), and marked as completed, being then 'moved' to the following task in the service description. Please, note the indication of the assembly context, and the recording of the execution information by the Observer Token object (tkOb), which will be used by the monitors.

Previous versions of this rule explicitly included the checks and actions associated to required changes in the system. In our new approach, however, monitors gather all the information on the metrics of interest, which will then be handled independently by corresponding observers. In this way, new observers may be added at any time without needing to modify these rules. With this modification, generic tokens were created to store the values obtained in the simulation and subsequently update the observers. This creates greater flexibility for the inclusion of new metrics for measuring non-functional properties, as it eliminates the need

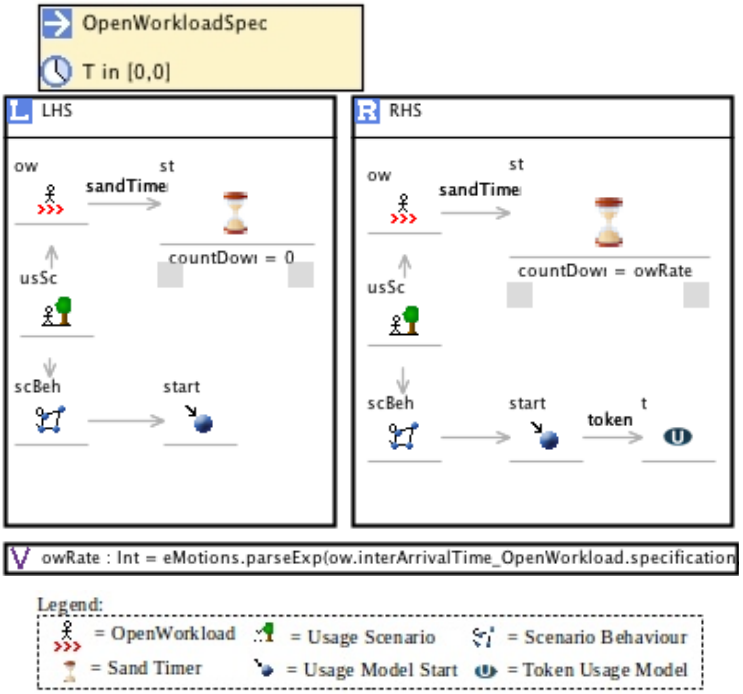



**FIGURE 8** OpenWorkloadSpec rule

to manipulate the rules specifying the behaviour of Palladio to change the requirements or adaptation mechanisms of specific systems. In summary, observer objects are in charge of collecting information on non-functional properties with which the performance analysis will be carried out by monitors.

## 5.2 | Adaptation Mechanisms

In this section, we describe how the adaptation rules can be modelled.[2] Although alternative specifications of these adaptation rules were presented in previous work,[?] the versions presented here exploit the possibility of separating the specification of the operations and the adaptation requirements, therefore making the approach more flexible, at the time that these rule become simpler and easier to understand and maintain.

The linkage between the rules specifying the adaptation operations and the QoS and SYBL rules of behaviour happens thanks to the SYBL and the SYBL Annotation classes. In adaptation rules, the adaptation Strategy is represented by objects of these classes.

Although other adaptation operations could be similarly specified, we focus here on what in Cloud Computing is known as vertical and horizontal scaling. Vertical scaling is the ability of resizing a server to increase or decrease its processing/storage capacity by adding or removing the amount of resources provided. Vertical scaling is limited by the amount of resources available. Horizontal scaling is the ability of a system to change resource capability by adding or removing nodes (i.e., instances or virtual machines), enabling the use of new resources, perhaps with better and larger processing/storage capacity, or releasing them if they are not further necessary.

Figures **??** and **??** show, respectively, sketches of the scale-up and scale-out adaptations on our running example — the corresponding SYBL annotation was shown in Listings **??** and **??**, in Section **??**. Figure **??** specifies a scale up operation on the number of replicas of the CPU resource available in the Server Node. Given $n$ replicas, its number can be increased in any amount $m$. The increment is specified as one of the adaptation parameters. A scale out consists on adding nodes, thus creating new containers and their resource specification and allocating a component on them. Figure **??** sketches the effect of a scale out operation on our running example. Given some number of DataBase nodes in some state of the system, each with a corresponding number of CPU resource replicas, the rule scheme represents the addition of a new application node, with the

[2]The complete set of rules is available, as the rest of the implementation, at https://www.scenic.uma.es/GTPAAS/.

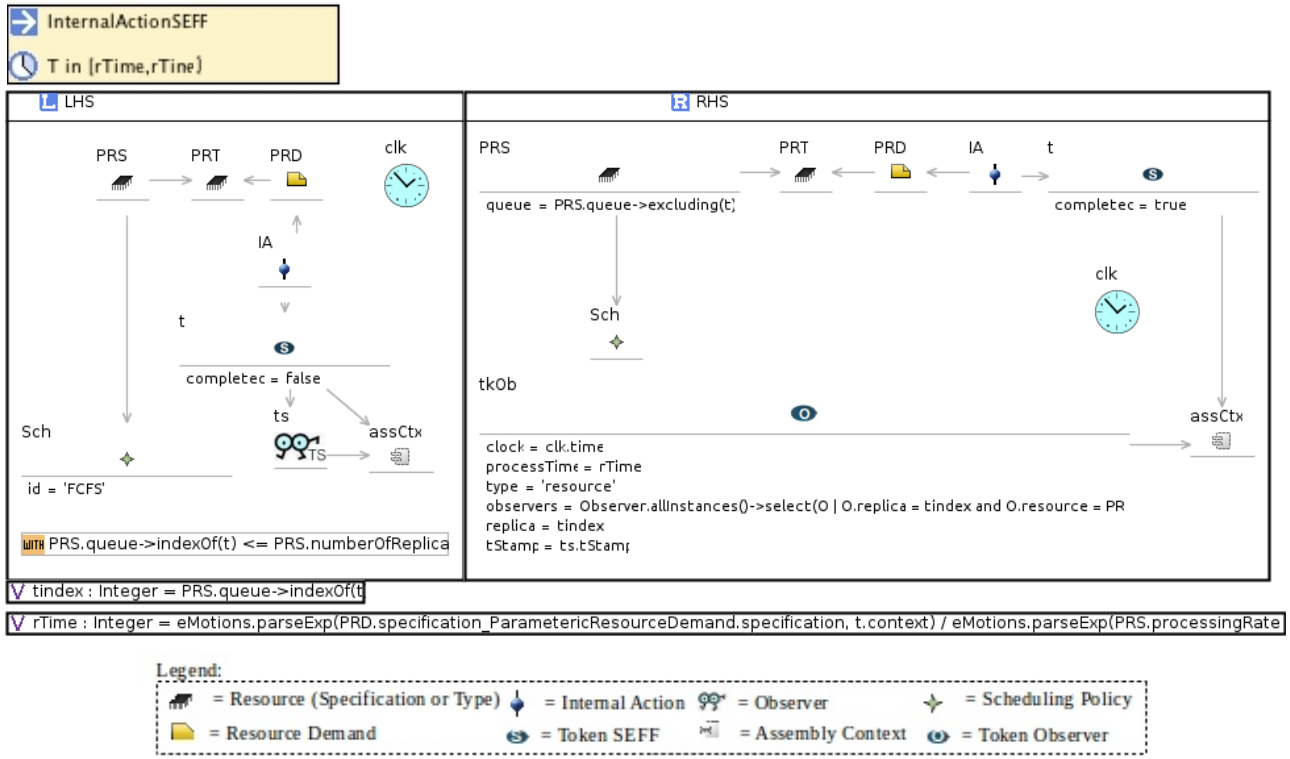



**FIGURE 9** InternalActionSEFF rule

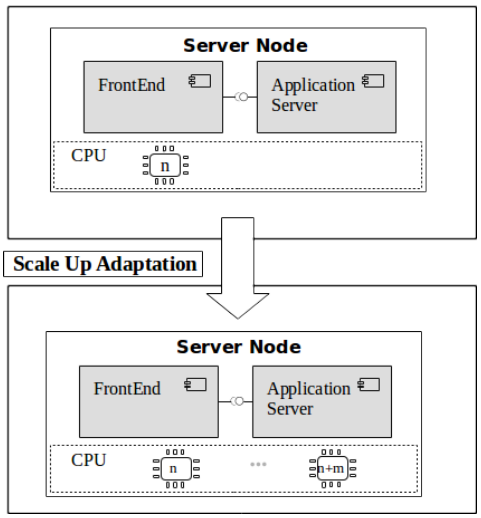



**FIGURE 10** Scale up adaptation scheme

specified initial number of replicas. Inverse operations — scale down to decrease the number of resource replicas and scale in to remove nodes — are also available.

Let us consider one of these operations in some further detail. Specifically, let us focus on the most challenging of them, the scale out associated to a DataBase Controller as the one in our example: when the average usage of CPU in the last time window goes over 65% a new node with the Database component has to be deployed. Although presented for our running example, the operation can be performed on any cluster of nodes upon the occurrence of the correspondent signal. Upon the validation of a SYBL annotation's constraint, a token command is released to trigger the action specified in the corresponding strategy.

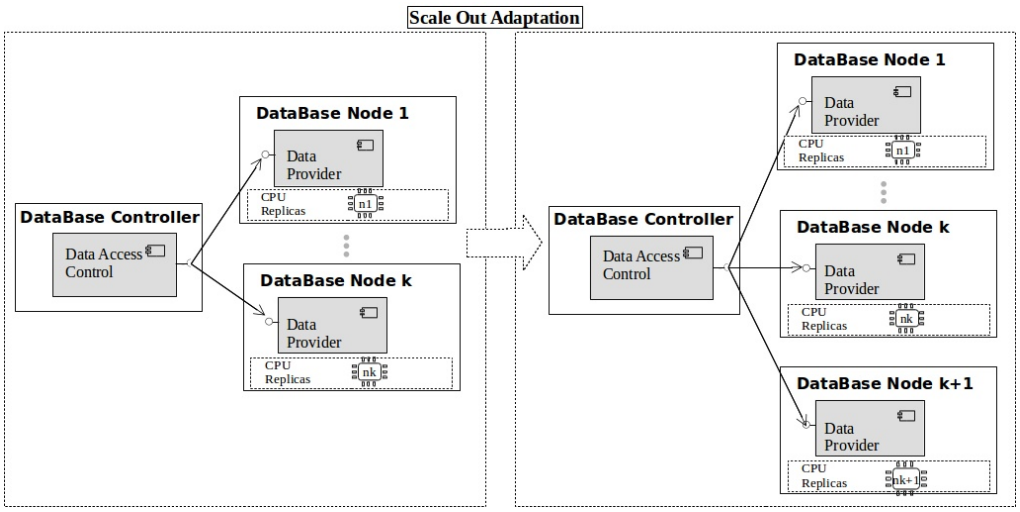



**FIGURE 11** Scale out adaptation scheme

Adaptation rules then operate on the structure defined in the Palladio models. The addition of a node in any state of the system will imply the modification of the models of the different views in the Palladio description:

- In the component model (see Figure **??**):
  - a Requires relation between the Controller component and the IDataBase interface should be created;
  - in the SEFF <control>, which models the control flow in the Controller component, a branch (with appropriate likelihood and corresponding external call action) should be created;
- In the assembly model (see Figure **??**):
  - an assembly context for the new node should be created along with its communication with the Controller component;
- In the allocation model (see Figure **??**):
  - an allocation of the DataBase component should be added to the new node.

Instead of having one single complex rule modelling what is sketched in the scale out adaptation depicted in Figure **??**, this operation is implemented by several e-Motions rules taking care of the different actions. The AddNode rule, depicted in Figure **??**, is the main one, triggering the addition of the node. Given an adaptation token linked to a SYBL annotation strategy with a ScaleOut action, a new node nNode is created in the environment recEnv, linked to the computing lan centre linkRes. In addition, a token indicating the creation of a new node tNnode will guide the triggering of subsequent rules, performing the rest of the necessary actions.

The scale out operation proceeds as follows:

1. As above explained, and depicted in Figure **??**, a scale out begins with the creation of the node in the AddNode rule.
2. Then, the NewNodeContext rule creates assembly and allocation contexts for the new node. The component for which the new context is created is indicated in the annotation of the specification, as well as the operation signature and the operation interface linked with it. When applying the rule, the new node token is linked with the new assembly context, the operation signature, and the operation interface, and receives the monitor list from the SYBL annotation.
3. The SpecResourceNewNode rule is responsible for configuring the specified resources, as defined in the initial model, for the new node. This rule is applied for each of the resources modelled in Palladio.
4. The ReqProvConnNewNode rule uses the assembly context, the operation signature and the operation interface to create the new assembly connector and the new operation required in the component that has the external action call. This component can be either a load balancer or a database controller. The external call is within a probability branch in a branch action.
5. The AddBranchNewNode rule adds a new probability branch, with the corresponding external call action, and the signature as indicated by the new-node token.
6. The IncBranchProbability rule balances the probabilities of all branches.

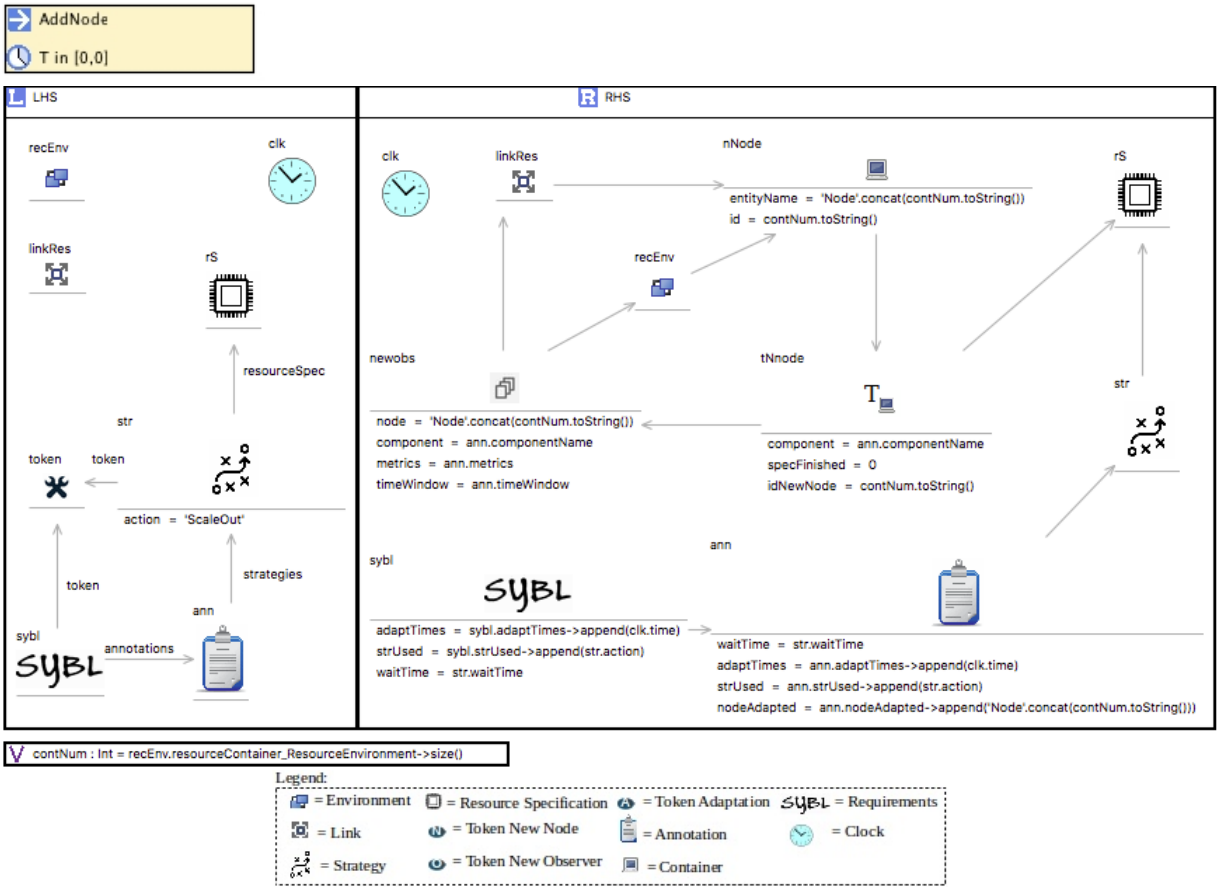



**FIGURE 12** AddNode rule

7. The CreatedRTOb and ResourceUsageOM rules create and link observers in accordance with what is specified in the monitors in the annotation.
8. The RemoveTokenNewNode rule is in charge of removing remaining operation tokens once all monitors have been appropriately handled.

The scale in operation proceeds as follows:

1. The DecBranchProbability rule disconnects a probability branch from the branch action, whose probabilities are rebalanced.
2. The RemoveBranch rule removes the probability branch object and the external action linked to it.
3. The RemoveNode rule eliminates the node along with all its connections and assembly context. This rule is triggered when all jobs present in the resource usage queue of the node to be eliminated have already been completed.
4. The RemoveObservers and RemoveTokenRemove rules remove all observers when it is certain that none still has any value that can be considered by a time window.

## 5.3 | Adaptations Guided by Non-Functional Requirements

Modelling the systems and the environments where they are going to be deployed is not enough in the presence of dynamic behaviour, once some kind of adaptation has to be modelled as well. Indeed, we need some way of specifying how the adaptation is managed and controlled. Although SYBL was not designed to be used for predictive analysis, but to connect different monitoring tools and cloud APIs, and to indicate elasticity mechanisms for different services, we consider it very appropriate for our purpose due to its ability to describe constraints, monitoring and adaptation strategies.

In previous works,[??] some adaptation mechanisms were defined as transformation rules on the model of the system under analysis. However, these works were limited to the use of the observers according to what was proposed by Troya et al.,[?] which

⟨*Constraint*⟩ := constraintName : CONSTRAINT **ComplexCondition**

⟨*Monitoring*⟩ := monitoringName : MONITORING varName=**MetricFormula**

⟨*Strategy*⟩ := strategyName : STRATEGY WHEN **ComplexCondition** : action(parameterList)
| strategyName : STRATEGY WAIT **ComplexCondition**
| strategyName : STRATEGY STOP
| strategyName : STRATEGY RESUME

⟨*MetricFormula*⟩ := metric | number | metricFormula **MathOperator** metric
| metricFormula **MathOperator** number

⟨*ComplexCondition*⟩ := **Condition | ComplexCondition BitwiseOperator Condition**
| (**ComplexCondition BitwiseOperator Condition**)

⟨*Condition*⟩ := metric **RelationOperator** number | number **RelationOperator** metric | **Violated**(name)
| **Fulfilled**(name)

⟨*MathOperator*⟩ := + | − | ∗ | /

⟨*BitwiseOperator*⟩ := OR | AND | XOR | NOT

⟨*RelationOperator*⟩ := <| >| >= | <= | == | !=

**FIGURE 13** BNF grammar of SYBL

means the use of metrics and adaptation specifications defined directly on the behaviour rules and linked to the object to be analysed. In order to make the definition of quality metrics even more flexible, the management of the non-functional properties is performed by a series of independent rules of behaviour.

In Section **??**, two SYBL annotations for our running example were presented. The grammar of the SYBL language, in Backus Naur Form (BNF), is shown in Figure **??**. However, to be able to use SYBL annotations textually specified using this grammar they must be transformed into models conforming to a SYBL metamodel, which is shown in Figure **??**. Classes in the SYBL metamodel represent the expressions and terms that the grammar defines. The `SYBLrequirements` class relates to one or more `REQUIREMENTSAnnotation` classes, has two attributes and indicates when an SYBL annotation has been violated or fulfilled. The `REQUIREMENTSAnnotation` class represents the built specification, as presented in Section **??**. Each annotation must contain the specification of the restrictions (`REQUIREMENTSComplexCondition`), the adaptation strategies (`REQUIREMENTSComplexStrategy`), the specification of the monitors and their monitoring metrics (`REQUIREMENTSMonitoring` and `REQUIREMENTSMetric`).

SYBL Monitors work in an integrated way together with observers to collect monitoring information on the different metrics being observed. Systems may then, depending on this gathered information, adapt in different ways by performing different operations, like scale up/down, scale in/out, etc. Since the collected monitoring information is stored in the Observer objects, direct access to current values, windows of values of certain length, and complete histories of data are provided through them. The monitoring infrastructure is integrated with the e-Motions observers and runs on the different levels: for each observer, a monitor is created, and a monitor can be associated to one or more observers (in the case of two or more replicas of resource, for example, one observer is created for each replica). Moreover, a monitor can monitor another one. The monitor which directly monitors an observer belongs to level one and the monitor of other monitors belongs to level two. The metamodel proposed along with the definition of behavioural rules allow any quality metrics to be inserted.

The specific instantiation of observers and monitors for our running example is depicted in Figure **??**. For inserting a new metric, we just have to specify it in SYBL and create, in the metamodel, a subclass Observer corresponding to it. The information about each metric is saved and calculated during simulation time using the monitors.

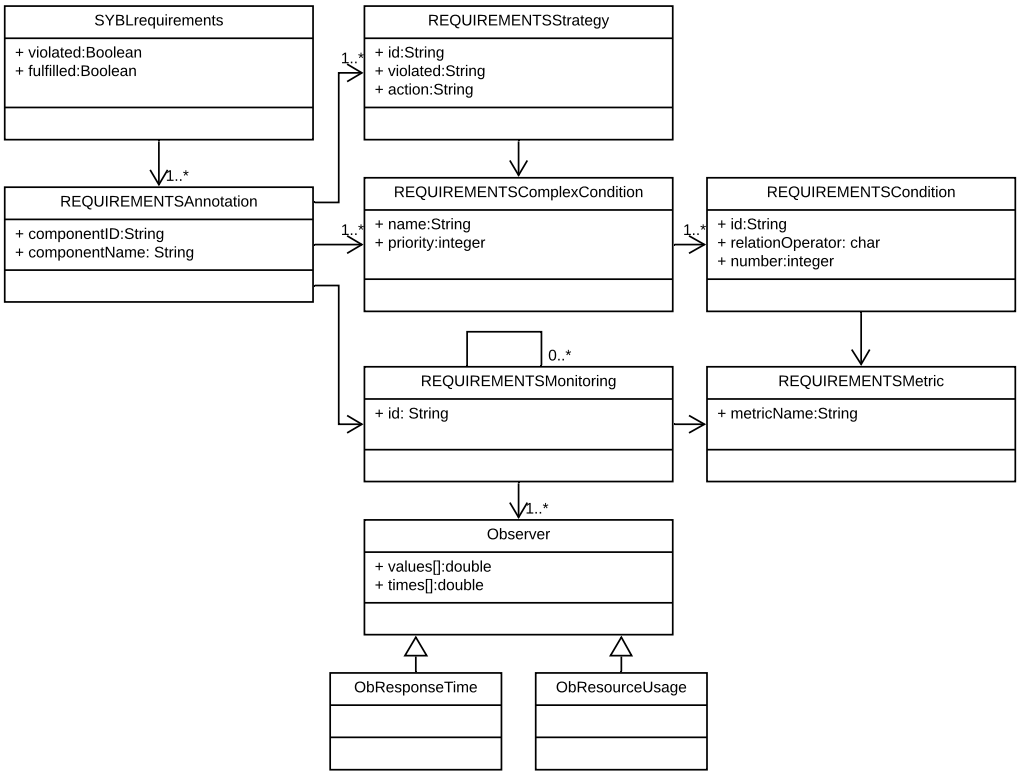



**FIGURE 14** Requirements Control Metamodel

The SYBL specifications, such as specified in Section **??** and written according to the metamodel of Figure **??**, are used as a reference for controlling non-functional requirements, which are evaluated using a set of rules integrated with the rules defined for the adaptation control.

The CheckConditions rule in Figure **??**, specifies the evaluation of the conditions associated to a specific SYBL annotation. The checkStatus and checkConstraint OCL functions[3] check the values of the restrictions established in the specification and compare with the values obtained during the simulation by the observers managed by the monitors. This rule checks whether a restriction is violated. If so, it creates a trigger for other rules to decide which adaptation strategy should be applied, according to specification. This is done by releasing tokens that control the flow of rules. The behaviour rules checks whether an adaptation action is necessary, and if so, it decides which strategy should be applied, according to specification, and releases a token that indicates the action to be executed.

The management of non-functional properties and the adaptation operations includes rules for the management of controllers, monitors, observers, and time windows. Figure **??** depicts the procedure. The beginning of the scheme indicates the first rule for the adaptation control and non-functional requirements. The `CheckConditions` rule continuously verifies the specified constraints. This verification consists in the comparison of the defined values in the SYBL specification to the values obtained during the monitoring in simulation time. The scheme indicates that the monitoring values are obtained from the observers, which are updated by tokens that were inserted in the Palladio rules. The update of the monitoring values occur following a pre-established time window.

If the `CheckConditions` rule identifies that a constraint was violated, the `AdaptationStrategy` rule is triggered and it will indicate which adaptation should be applied, according to the SYBL specification. The waiting time between adaptations is also pre-established. Thus, if the time between adaptations overcomes the waiting time a token is created and it will trigger the corresponding adaptation rule to the specified adaptation. The impact of the adaptation actions will be discussed in the next section.

[3] OCL (Object Constraint Language) auxiliary functions can be defined in e-Motions using e-Motions Helpers.

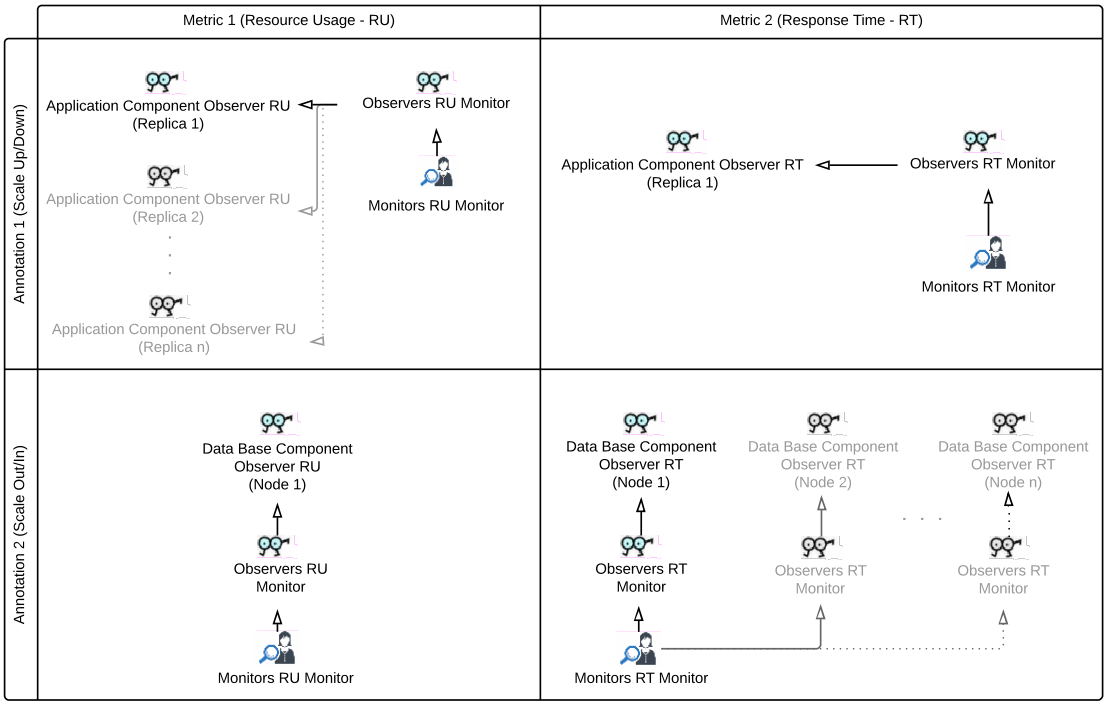



**FIGURE 15** Relations between observers and monitors

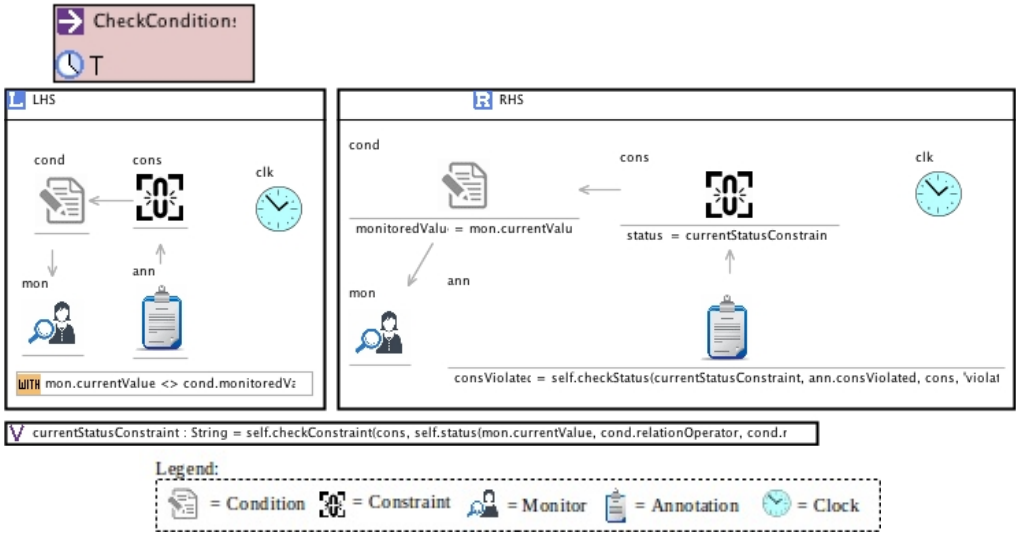



**FIGURE 16** CheckConditions rule

The control of the evolution of the systems can be seen as a typical MAPE-K (Monitor-Analyse-Plan-Execute over a shared Knowledge) feedback loop.[?] As illustrated in Figure **??**, in our case, the knowledge is represented by the set of specifications, models and behaviour definitions — the `Palladio Rules`, the `Palladio Model`, the `Adaptations` rules, the `Non-Functional Requirements` specification, and the `Control` rules. In summary, the monitoring is carried out with the information obtained by the tokens in the Palladio rules, and updated in the observers. The values obtained from the monitoring

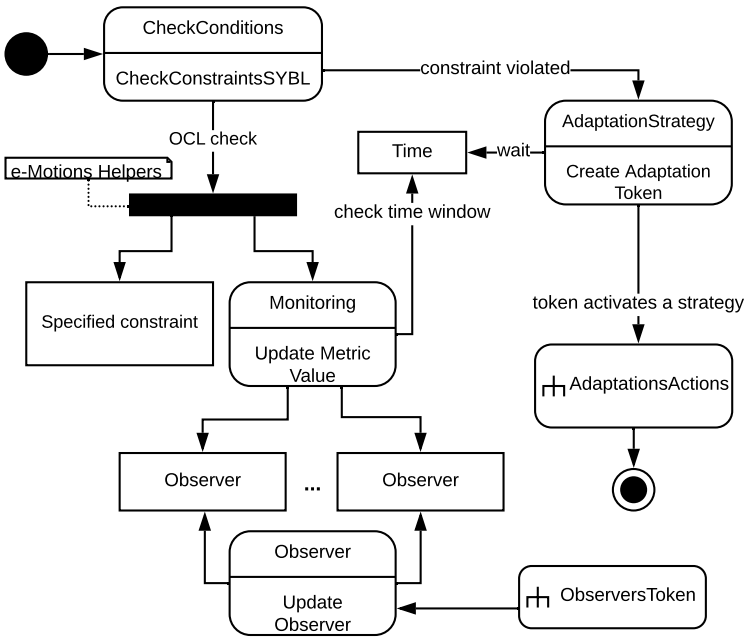



**FIGURE 17** Controller scheme

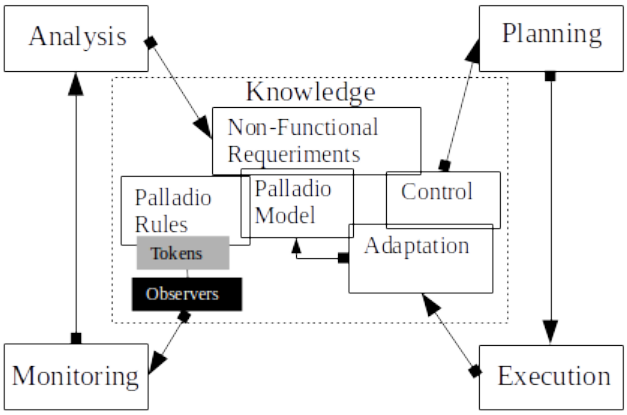



**FIGURE 18** A MAPE-K Loop interpretation of the approach

are often analysed by comparing them with the specified values of non-functional requirements. The control rules trigger the adaptation planning and, if necessary, they are executed when possible.

We present in the next section the analysis phase, which proposes the adjustment in knowledge and the re-simulation as a process for analysing models of self-adaptive systems.

## 6 | PERFORMANCE ANALYSIS

Once the system is specified, we can carry on its performance analysis. As explained in Section **??**, the Palladio model, the adaptation operations, and the SYBL annotations are used to generate a Maude specification which is then used for simulation. From these executions, the corresponding data is collected and charts are produced to gather insight on the system and the adaptation strategies.

We show in this section some results on the analysis of our running example, showing how the information obtained on our system may help us in the decision taking, both on the architecture of the system and on the elasticity specifications. We first summarize the parameters used on the simulations and then discuss the results provided in three different scenarios on our running example, namely a non-adaptive case, and systems with the SYBL annotations in Listings **??** and **??**, respectively. We will refer to these annotations, respectively, as Ann1 and Ann2. In all three cases, we use the Palladio model presented in Sections **??** and **??** as initial model, and the adaptation operations introduced in Section **??**.

Notice that the annotations that we use here are acting on different components, using specific metrics. In practice, we may use multiple annotations simultaneously, acting on different components, and the insights may lead us to further alternatives. For space reasons, however, here we show just a small set of results, emphasising on the type of results we get and how to interpret them, instead of actually trying to optimise the specification of a complex system. Additional examples may be found on the tool's web site.

We have seen in previous sections that we may use different metrics for our systems. We use three different metrics in the discussion in this section. Two of them were introduced in Section **??**, resource usage and response time, as they were used in the specification of the SYBL annotations for our example. However, different metrics may also be considered for the analysis of systems. In this section we show how the number of tasks waiting to be serviced by each of the components may help us to understand its behaviour.

## 6.1 | System with no Adaptation

As a first experiment for our example, we need to understand its behaviour without any kind of adaptation, to view the problems in the model configuration, as for example the possibility of bottlenecks causing delays or if there is a problem related to the use of resources in the node where one of the components is allocated.

First, we need to add a few additional parameters and assumptions. In our example, three nodes were specified, and to simplify both the presentation of the model and of the results, we consider that the nodes on which the components are allocated have the same configuration. Each of these nodes is specified with CPU processing rate of 1000, First-Come-First-Serve scheduling, and initially there is one CPU resource replica for each of them. We assume that the communication network as ideal, i.e., as having no flaws nor delays. In other words, the delay value is specified to be 0, and the bandwidth infinite.

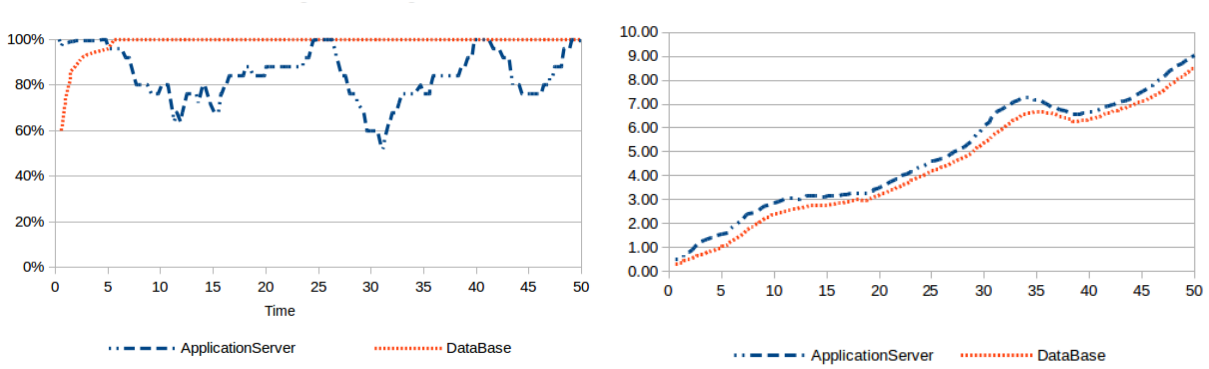
In the example, the ApplicationServer and DataBase components demand, respectively, 200 and 300 CPU units. The demand of CPU of a specific service and the processing rate of the node on which it operates give us its runtime. For instance, the service of the ApplicationServer component — CPU resource demand 200 — is executed on a CPU with processing rate 1000, which means that its execution takes 0.2 t.u. Notice that due to their specificities, the FrontEnd and Controller components were modelled with no CPU demand. Hence, we carry out the analysis of the quality metrics regarding the impact of the workflow evolution on the ApplicationServer and DataBase components.

To understand the system with no adaptation, we first need to understand how the runtime of each component (and the total runtime) is calculated. We use the Resource Usage (RU) and the Response Time (RT) metrics to obtain information on the ApplicationServer and DataBase components. The RU metrics will be obtained from the monitoring of the use of the nodes where each of these components is allocated. As we have not modeled any other type of resource demand, and the communication channel is ideal, the RT of the ApplicationServer component will indicate the total system execution time, and the RT of the DataBase component will indicate the time it takes for a job from its arrival to the completion of data processing.

As specified in the usage model in Section **??**, we use a closed workflow with inter-arrival time provided by the stochastic expression Exp(5.0), i.e., tasks arrive following an exponential probability distribution with rate parameter 5.0 t.u.

The observation of the metrics on the simulation of the system with these parameters produces the charts in Figure **??**. Regarding resource usage (Figure **??**), we observe that the DataBase component's node goes to using 100% of its resources 5 t.u. after the beginning of the simulation. The usage varies in the node of the ApplicationServer component in the range 60-100% for most of the time. In both cases, the response time remained increasing throughout the simulation (chart in Figure**??**).

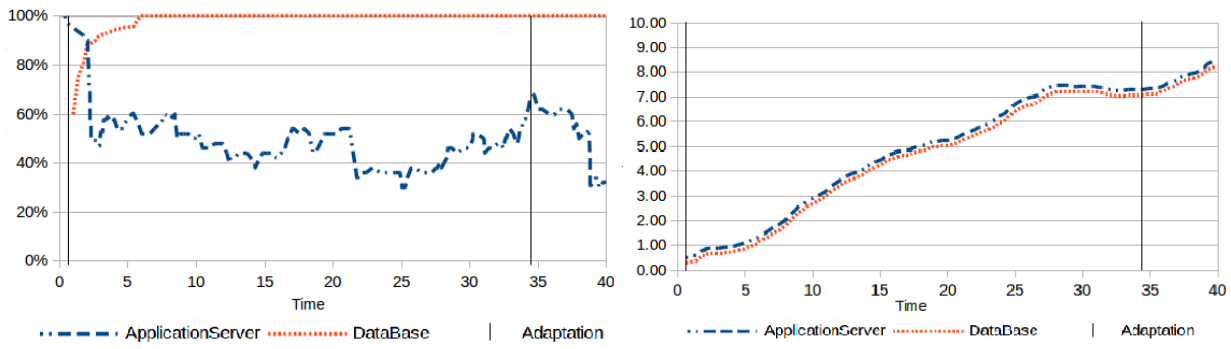
These charts tell us that the response time degrades along time, and possibly due to an under allocation of resources, perhaps in the node of the ApplicationServer component, but for sure in the DataBase component's node.




**(a)** Resource usage without adaptation actions

**(b)** Response time without adaptation actions

**FIGURE 19** Resource usage and response time without adaptation actions




**(a)** Resource usage

**(b)** Response time

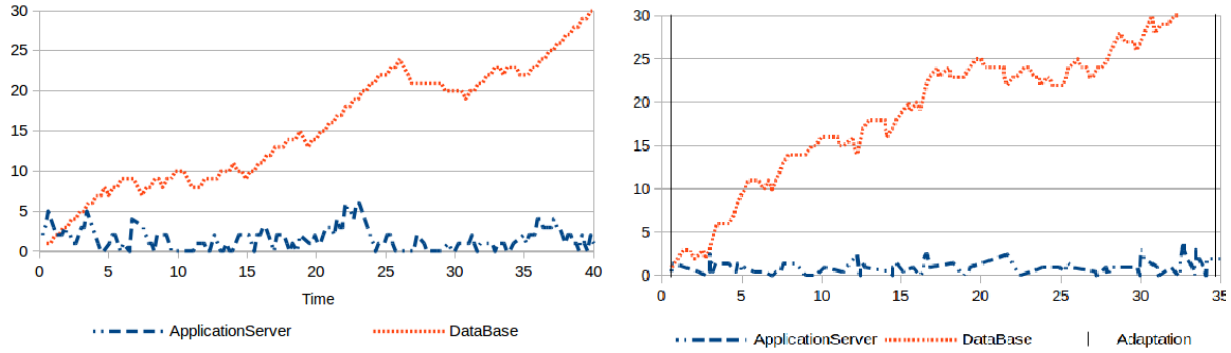
**FIGURE 20** Resource usage and response time with Annotation Ann1

## 6.2 | Analysis with Ann1: Resource Usage of the ApplicationServer's Node

Annotation Ann1 (Listing **??**) specifies an adaptation on the ApplicationServer component considering the resource usage of the node where the component is allocated. Specifically, the adaptations occur when the average in the resource usage monitor is above 65% (scale up) and below 30% (scale down).

Figure **??** depicts the resource usage in components ApplicationServer and DataBase in a simulation with the Ann1 elasticity specification. The vertical lines in the charts show the times at which the adaptation operations get triggered. Thus, we observe a scale up adaptation at time 0.4. Right after that time, the resource usage decreases considerably for the ApplicationServer component, varying between 30% and 60% for most of the simulation time. However, the change occurred only in the ApplicationServer component's node, which explains the change in its resource usage, while the node where the DataBase component was allocated stayed with a resource usage of 100% during the most part of the simulation. Therefore, despite the improvement, the response time values of both components stayed as the ones presented in the simulation without adaptations (see Section **??**), which can be explained by the fact that the ApplicationServer component depends on the DataBase component to finish its execution. I.e., not improving the response time of the DataBase component means that the response time of the ApplicationServer component will not be altered.

As the value of resource usage metrics is obtained from monitoring the use of the nodes on which each of these components is allocated, and as we use FCFS scheduling for the resource usage, this value considers the time in the resource usage queue plus the processing time (resource demand / node processing rate), the monitoring of the resource usage queues may bring some insight. Figure **??** shows that the resource usage queue of the node of the ApplicationServer component remains most of the time below 5, both in the non-adapted scenario and in the adapted scenario operated by Ann1. In contrast, the resource usage queue




**(a)** Resource queue without adaptation actions

**(b)** Resource queue with Ann1

**FIGURE 21** Resource queues on the nodes of the components without adaptation and with Annotation Ann1

of the node that allocates the DataBase component remains growing in both scenarios, which shows a bottleneck. It indicates that, to improve the response time of our application, adaptations that focus on the DataBase component should be a priority.

## 6.3 | Analysis with Ann2: Resource Time of the DataBase's node

Annotation Ann2 specifies an adaptation on the DataBase component considering its response time, which refers to the delay time of a work to be concluded by a component, from its request until its response. In accordance to the Ann2 specification, the adaptations occur when the average in the response time monitor is below 0.2 t.u. (scale in) or above 0.5 t.u. (scale out).

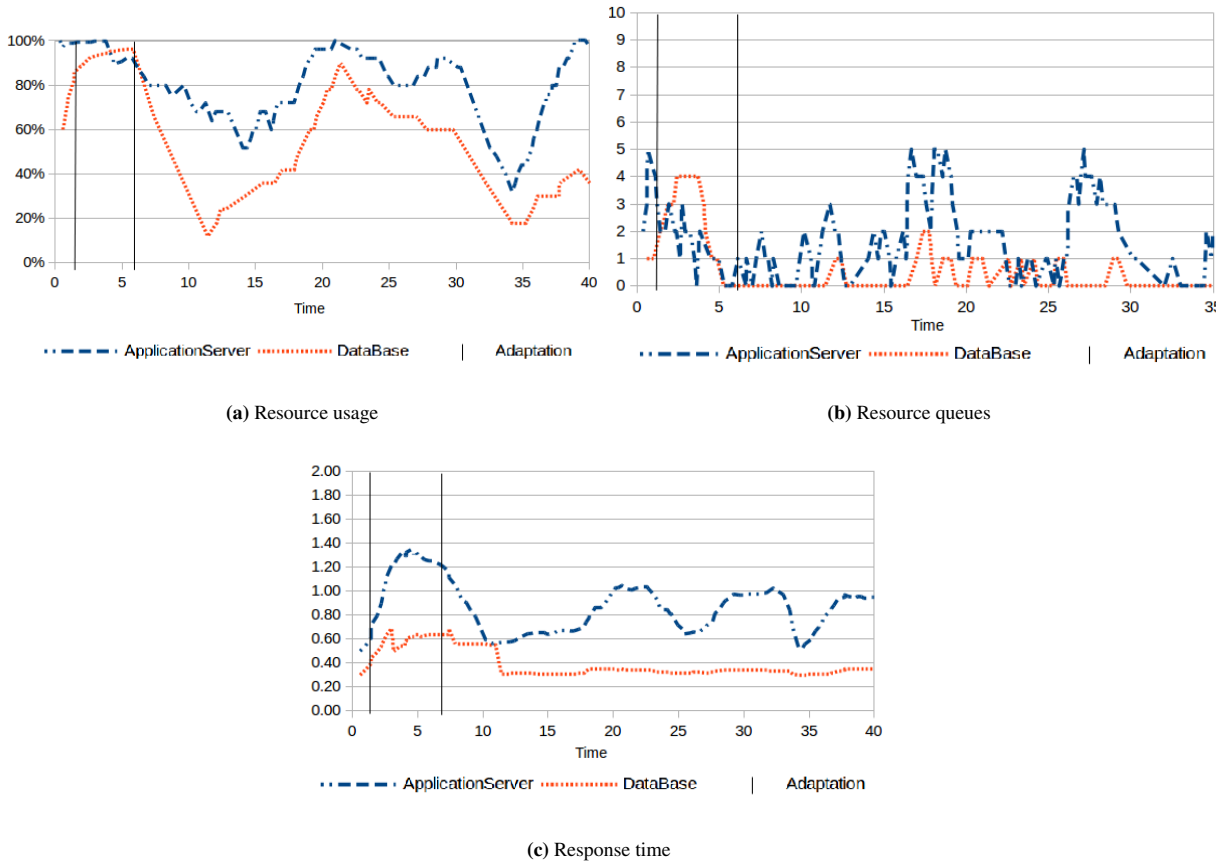
Figure **??** shows the response time average of the components ApplicationServer and DataBase in a simulation with the Ann2 adaptation specification. We observe the application of scale out adaptation operations at times 2.3 and 7.3. After these operations, the response time for both components decreased significantly, stabilizing with the ideal average of 0.3 t.u. for the DataBase component and varying between 1 t.u. and 0.5 t.u. for the ApplicationServer component. The adaptation occurred only on the DataBase component, nevertheless, we notice the impact in the whole system. The impact in the resource usage in both components was also observed, as depicted in Figure **??**. The resource usage average in the nodes that allocate the DataBase component dropped below 60%. Despite the lack of stability, the results are promising, and show that an increase in the number of nodes that allocate the components offers a significant improvement in both quality metrics. When looking at the resource queues, we observe a decrease for both components: the ApplicationServer component presented an average of two works in the queue and the DataBase component presented an average of one work in the queue (Figure **??**).

## 6.4 | Conclusions from our Analysis

Although there has been an improvement in the response time using Annotation Ann2 (Section **??**), the response time of the application averaged 0.89 t.u., which is above what is specified as a non-functional requirement. To correct this, we have specified a third annotation in which, in addition to the scale out adaptation in Ann2, which inserts new nodes with instances of the DataBase component, operates scale-up adaptations on the DataBase component. Figure **??** shows the results obtained with the adjusted annotation. As we can observe, it brings an earlier stability in the response time of the DataBase component, and an average of 0.71 t.u. in the application response time. The system with this SYBL annotation now satisfies the given requirements.

The analysis of the system concludes with some modification suggestions. Specifically:

- The scale up adaptation strategy is added on the node where the DataBase component is allocated when the CPU usage is greater than 65%.
- Adaptations in the node in which the DataBase component is allocated have priority over adaptations in the node in which the ApplicationServer component is allocated.
- Considering the average CPU usage obtained in the experiment with the adjusted specification, the new limit for the use of CPU on the nodes where DataBase component is allocated is 68%.




**(a)** Resource usage

**(b)** Resource queues

**(c)** Response time

**FIGURE 22** Resource usage, resource queues and response times with Annotation Ann2

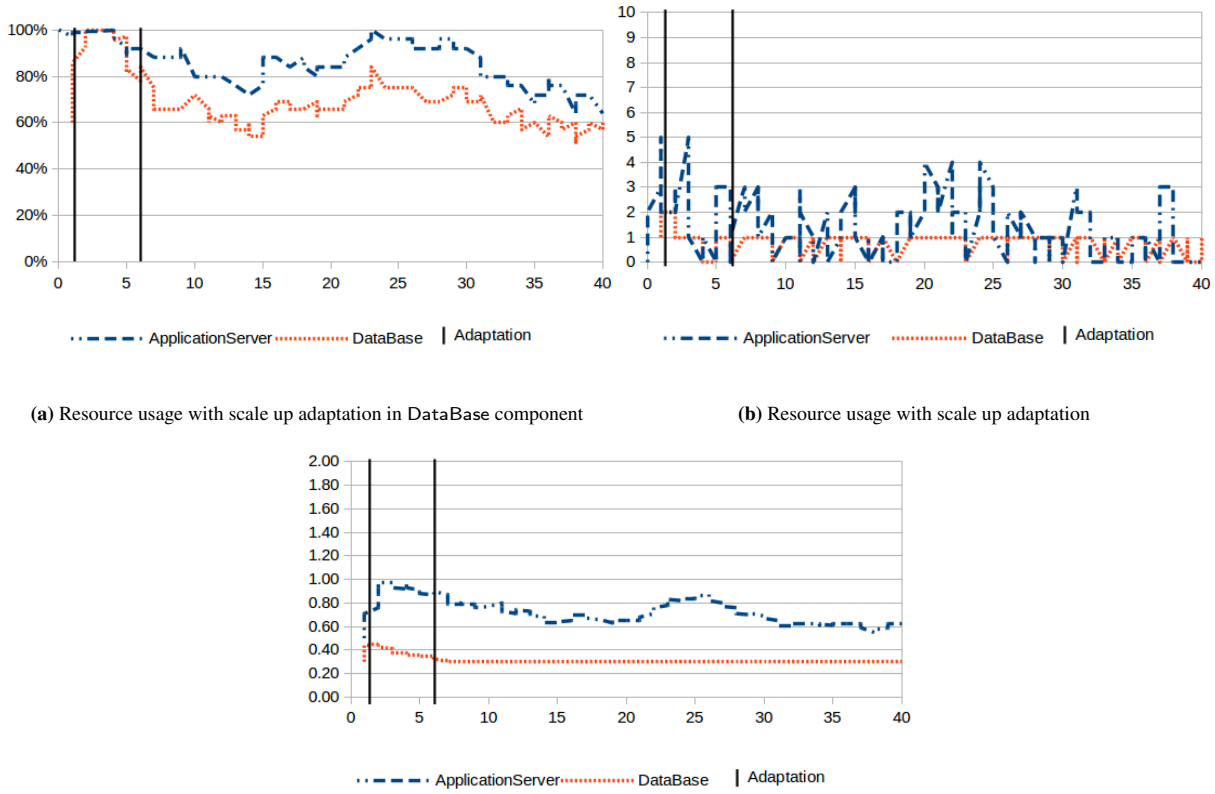
- Considering the average response time obtained in the experiment with the adjusted specification, the new response time limit for the application is 0.71 t.u.

The simulations showed that, regarding the change in the initial model, two increments in the number of replicas in the CPU resource of the nodes were sufficient in both cases (ApplicationServer and DataBase). These adaptations, combined with at least one more node, will bring good quality of service results for the example presented here.

## 7 | RELATED WORK

Different approaches have been proposed for analysis of self-adaptation systems, including techniques and tools based on queue networks,[?] Petri nets,[?] stochastic models,[?] Markov chain,[?] and graph transformation.[?]

Becker et al. propose in SimuLizar[?] a model-oriented approach to model self-adaptive systems and analyse the performance of its transitory phases. It is developed as an extension of Palladio for the performance analysis of self-adaptive systems at design time. The simulation scope is limited to only a set of rules that are triggered between the static environment models, which prevents it from testing multiples possible reachable states of systems in simulation time. SimuLizar was extended by Becker et al.[?] to support scale up/down adaptations, specifically the change in the number of replicas and the computation capacity of a node. Krachand and Scheerer proposed a later extension SimuLizar NG (New Generation),[?] with the goal of facilitating domain-specific extensions, and allowing adaptations to the model interpretation semantics through the reactive simulation in demanding scalability scenarios. As part of the Descartes project, which also uses Palladio's PCM for their design time phases, Huber et al.[?] propose a DSL to describe the behaviour of self-adaptive systems based on strategies, tactics and actions. It focus on runtime performance analysis, not in predictive analysis of applications at design time.




**(a)** Resource usage with scale up adaptation in DataBase component

**(b)** Resource usage with scale up adaptation

**(c)** Response time with scale up adaptation in DataBase component

**FIGURE 23** Resource usage, resource queues and response times in both ApplicatiomServer and DataBase components with scale up adaptation in DataBase component

In order to mitigate the difference in static model development in contrast to dynamism during the operations, Heinrich's iObserver project[?] proposes a metamodel that allows the reuse of development models during the operation phase. The approach aims to bridge the differences between the component-based design and its implementation, design and application configuration, and static and dynamic content. Even though the project recognises the importance of closer development models with the characteristics of dynamic systems, such as cloud systems, it does not use a dynamic model approach. Indeed, iObserver integrates design-time models, code generation, monitoring, analysis and run-time model update.

Grassi et al. proposed D-Klaper[?] as a tool for model-driven performance engineering that can be applied to self-adaptive systems. It uses an intermediate language to provide software design models, which can then be analysed. However, the D-Klaper language does not support the modelling of adaptation rules, nor the transformation of input models. The work by Falkner et al.[?] proposes an approach where the performance prediction is made at the beginning of the life cycle. To do this, workloads are modelled by estimating the resource consumption, capturing the CPU, memory and disk requirements. However, although they use models to represent the system, they are used to generate executable code for specific hardware and middleware deployments, and the results of the executions are presented to the expert through specific context views indicating whether the design meets the performance requirements, and cannot be considered a fully design time approach.

Abuseta and Swesi[?] propose an assistance tool for software adaptive-system developers in order to reduce development time and ensure robust functionality of the developed system. The framework is a reusable process of elements and system components to simulate and test the activities that exist in the self adaptive systems, and aims at increasing the developers' confidence on the robustness of their designs prior to their deployment.

Johnsen et al.[?] propose a model-based prediction approach to compare the effect, in terms of performance and accumulated cost, of selecting different instance types of virtual servers from Amazon Web Services (AWS). In order to do this comparison they use a highly configurable modelling framework for applications running on Apache YARN, the ABS-YARN, using the

executable semantics of Real-Time ABS, defined in Maude, as a simulation tool. However, they do not model the application but instead use values obtained from real measurements.

There are other approaches that propose different solutions for evaluating non-functional properties using models, such as the one by Bernardi et al.,[?] who propose the assessment of non-functional properties using multi-formalism approaches, but they do not address self-adaptive systems. In particular, they are proposing an integrated model to combine performance and reliability models, which allows the construction of models for performance analysis, and automated construction of multi-formalism models with less effort. Our purpose follows a different path, integrating models and languages in the e-Motions framework, which allows the performance analysis of non-functional properties, specifically, on self-adaptive systems, thus introducing an additional degree of advancement in the challenge of assessment non-functional properties of systems models.

## 8 | DISCUSSION OF THE APPROACH

We have presented a design-time approach that allows us to adjust and remodel our system specifications, their models, and even the behaviour of systems, and can lead to performance prediction once the results presented in the design-time analysis are validated in real application runs.

Our approach allows conducting analysis of models of self-adaptive systems. This article clarifies several points to be considered in order to conduct modelling and analysis of models at design time that consider the dynamism of self-adaptive systems, that is, the need for dynamic change in the execution of analysis of models, and the need for re-adjustments and re-analysis of the models executed. To proceed with these steps, a structure capable of offering the possibility of modifying each stage of the design of this type of system is necessary.

Building adaptation rules and a metamodel that is capable of acting on the simulation of self-adaptive systems is not a trivial task: it requires time and development effort. However, once done, it can contribute to the advance of research in the area, with the structure and initial modelling available. It is necessary to thoroughly analyse the behaviour to be modelled and identify the elements necessary for the construction of each metamodel. This task can only be performed from a detailed survey of functionalities and real behaviour, either to model a system, such as Palladio, or to model any type of adaptation, action or language that you want to include in a simulation.

This proposal does not seek to model a complete or even complex self-adaptive system, since this is not a trivial task to be addressed at design time either. The presented approach seeks to provide ways to conduct the analysis of system components and, for its progress, it is necessary to overcome technological limitations still present. The work presented here has some limitations that still need to be overcome, mainly originated from the trade-off between expressivity and efficiency. In this work we have tried to maximise expressivity. The e-Motions approach leaves a significant part of the computation on the conditions of rules. Although this greatly simplifies the rules and their transformation, it results in a high computational price during the simulations. Given the way in which these rules are executed, a significant number of rule matches are discarded when evaluating such conditions. This may hamper the scalability of the proposal, and will need to be overcome if the tool is to be applied on complex systems. This problem was mitigated in this work by creating the rules with the smallest possible number of similar objects, in order to reduce the number of matches. However, a more general solution should be found if we are looking for user-defined annotations and adaptation rules.

Although the approach, like others at design time, cannot be applied to a context such as that found in a real scenario, it still represents a significant contribution and a qualitative gain for practical applications. Considering that, by anticipating instruments and mechanisms of analysis and adaptation in design time, it makes it possible to: (1) establish an understanding of the changing nature of self-adaptive systems through the modelling of services and their adaptations, also enabling the simulation of these models; and (2) the establishment of guidelines capable of assisting in the construction of systems, facilitating the set of decisions at design time, such as specification of requirements that consider restrictions, monitoring and adaptation strategies, as well as the modelling of the system and its behaviour, and design-time analysis to build a model with better quality of service results.

The results of the simulations using our procedural and flexible approach showed us that it is possible to incorporate changes in models of adaptive systems and, mainly, that our approach is useful to model and measure the unpredictability of these systems.

# 9 | CONCLUSIONS AND FUTURE WORK

Designing software systems, which are becoming increasingly larger and more complex, has become a challenge due to the need to take into account not only the overall structure of the systems themselves, but also their allocation, functionality, and communications of their components. In this regard, the Software Engineering community should concentrate efforts on building approaches that aim to establish the systems' architectures for their implementation, but that are also suitable to the analysis of their quality attributes, such as performance.

When considering the scalability, elasticity and adaptability of systems, dynamism and autonomy are challenging requirements, and considering them at design time is still a difficult task. Emerging behaviours and opportunistic interactions cannot yet be predicted at design time. However, these efforts should focus mainly on solutions related to the establishment of models that understand the changing nature of these systems in order for projects to model services and its adaptations efficiently. These models must be tested so that it is possible to find guidelines capable of assisting in the construction of systems so that they mitigate delays and failures, and that do not directly impact the users' experience. These guidelines concern a set of decisions at design time, such as the specification of requirements that consider constraints, monitoring and adaptation strategies, as well as the modelling of systems and their behaviour, and the system analysis at design time in order to build models with better quality of service results.

This paper presented a proposal that uses Palladio, e-Motions, Maude and SYBL in such a way that it enables expressiveness and flexibility in the specification, modelling, behavioural definition and performance analysis of self-adaptive models using a procedural approach. Such procedural organisation has helped us to understand the formulation of each of the steps, which makes the approach flexible and consistently reproducible.

We model adaptation mechanisms as generic adaptation rules. We have illustrated our approach by modelling in/out scale and up/down scale rules, triggered in response to breaches of restrictions. We specify the elasticity requirements of the systems, allowing their adjustment. The verification of the specification and the adaptation in different components in the system allowed us to verify that previously specified strategies may be misleading and our approach proposes that the readjustment and analysis be carried out at design time.

The procedure and the practical experience have allowed us to validate our hypothesis. The initial specifications, model and adaptations must be tested in simulation time, and the results obtained through their use during a simulation could be submitted for re-simulation and re-analysis, which would lead us to a more precise diagnosis of the system and, eventually, to a re-adjustment towards better results. This was possible thanks to the possibility of modifying the model during simulation time, the facilitation of the writing of the requirements specifications (constraints, monitoring and strategies) using a simple language like SYBL, and the subsequent analysis of the impact of the adaptations on the system.

As future work we will model the communication channels to consider the impact of communication networks on systems. The current tool can model the behaviour of Software-Defined Networks (SDN) or even the Virtualization of Network Functions (VNFs), but the field itself still has challenges to be overcome, such as the pursuit of ways to model an SDN that interacts with the model of a system in order to represent minimal behaviour of a real network, or even, to model the behaviour of a VNF in the best possible way, since this is an NP-difficult problem and several proposals can be presented.

Finally, we intend to advance the analysis of this approach by integrating the characteristics of Cloud, Fog and Edge Computing to create a model to simulate Smart Cities scenarios, presenting a proposal for modelling SDNs and VNFs. For this, it will be necessary, mainly, to model the communication network and the components that have characteristics of IoT devices to be used together with the techniques and tools presented in this paper.

## Acknowledgements

This work has been partially supported by Ministry of Science and Innovation and FEDER projects PGC2018-094905-B-100 and UMA18-FEDERJA-180, by Universidad de Málaga, Campus de Excelencia Internacional Andalucía Tech, and by funding agency CNPq of the Ministry of Science and Technology (MCT), Brazil.

## References

1. Koziolek H. Performance evaluation of component-based software systems: A survey. *Perform. Eval.* 2010; 67(8): 634–658.

2. Balsamo S, Marco AD, Inverardi P, Simeoni M. Model-Based Performance Prediction in Software Development: A Survey. *IEEE Trans. Software Eng.* 2004; 30(5): 295–310.

3. de Sousa AO, Bezerra CIM, Andrade RMC, Filho JMSM. Quality Evaluation of Self-Adaptive Systems: Challenges and Opportunities. In: Machado I, Souza R., eds. *XXXIII Brazilian Symposium on Software Engineering (SBES)*ACM. ; 2019: 213–218

4. Criado J, Martínez-Fernández S, Ameller D, Iribarne L, Padilla N, Jedlitschka A. Quality-aware Architectural Model Transformations in Adaptive Mashups User Interfaces. *Fundam. Inform.* 2018; 162(4): 283–309. doi: 10.3233/FI-2018-1726

5. Criado J, Martínez-Fernández S, Ameller D, Iribarne L, Padilla N. Exploring Quality-Aware Architectural Transformations at Run-Time: The ENIA Case. In: Bellatreche L, Pastor O, Almendros-Jiménez JM, Ameur YA., eds. *6th International Conference on Model and Data Engineering (MEDI)*. 9893 of *Lecture Notes in Computer Science*. Springer; 2016: 288–302

6. Chen B, Peng X, Liu Y, Song S, Zheng J, Zhao W. Architecture-Based Behavioral Adaptation with Generated Alternatives and Relaxed Constraints. *IEEE Transactions on Services Computing* 2019; 12(1): 73-87.

7. Edwards R, Bencomo N. DeSiRE: Further Understanding Nuances of Degrees of Satisfaction of Non-functional Requirements Trade-Off. In: Andersson J, Weyns D., eds. *IEEE/ACM 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*; 2018: 12-18.

8. Bezerra CIM, Andrade RMC, Monteiro JMS, Cedraz D. Aggregating Measures Using Fuzzy Logic for Evaluating Feature Models. In: Capilla R, Lochau M, Fuentes L., eds. *12th International Workshop on Variability Modelling of Software-Intensive Systems*ACM. ; 2018: 35–42

9. Serral E, Sernani P, Dalpiaz F. Personalized adaptation in pervasive systems via non-functional requirements. *Journal of Ambient Intelligence and Humanized Computing* 2017; 9. doi: 10.1007/s12652-017-0611-4

10. Franco JaM, Correia F, Barbosa R, Zenha-Rela M, Schmerl B, Garlan D. Improving Self-Adaptation Planning through Software Architecture-Based Stochastic Modeling. *J. Syst. Softw.* 2016; 115(C): 42–60. doi: 10.1016/j.jss.2016.01.026

11. Sanislav T, Mois G, Miclea L. An Approach to Model Dependability of Cyber-Physical Systems. *Microprocessors and Microsystems* 2015; 41. doi: 10.1016/j.micpro.2015.11.021

12. Bezerra CIM, Barbosa J, Holanda J, Andrade R, Monteiro J. DyMMer: a measurement-based tool to support quality evaluation of DSPL feature models. In: Mei H., ed. *20th International Systems and Software Product Line Conference (SPLC)*ACM. ; 2016: 314-317

13. Raibulet C, Arcelli Fontana F, Capilla R, Carrillo Sánchez C. An Overview on Quality Evaluation of Self-Adaptive Systems. In: Mistrik I, Ali N, Kazman R, Grundy J, Schmerl B., eds. *Managing Trade-offs in Adaptable Software Architectures*Morgan Kaufmann. 2017 (pp. 325-352)

14. Happe J, Koziolek H, Reussner R. Facilitating Performance Predictions Using Software Components. *IEEE Software* 2011; 28(3): 27–33.

15. Becker S, Koziolek H, Reussner R. The Palladio component model for model-driven performance prediction. *Journal of Systems and Software* 2009; 82(1): 3–22. doi: 10.1016/j.jss.2008.03.066

16. Becker M, Becker S, Meyer J. SimuLizar: Design-Time Modeling and Performance Analysis of Self-Adaptive Systems. *Software Engineering* 2013; 213: 71–84.

17. Rivera JE, Durán F, Vallecillo A. A graphical approach for modeling time-dependent behavior of DSLs. In: DeLine R, Minas M, Erwig M., eds. *IEEE Symp. on Visual Languages and Human-Centric Computing (VL/HCC)*IEEE. ; 2009: 51–55.

18. Copil G, Moldovan D, Truong HL, Dustdar S. SYBL: An extensible language for controlling elasticity in cloud applications. In: Balaji P, Epema D, Fahringer T., eds. *13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*IEEE. ; 2013: 112–119.

19. Moreno-Delgado A, Durán F, Zschaler S, Troya J. Modular DSLs for flexible analysis: An e-Motions reimplementation of Palladio. In: Cabot J, Rubin J., eds. *10th European Conference on Modelling Foundations and Applications, ECMFA*. 8569 of *Lecture Notes in Computer Science*. STAF. Springer; 2014: 132–147.

20. Becker S, Koziolek H, Reussner RH. Model-Based Performance Prediction with the Palladio Component Model. In: Cortellessa V, Uchitel S, Yankelevich D., eds. *6th Intl. Workshop on Software and Performance (WOSP)*ACM. ; 2007: 54–65.

21. Clavel M, Durán F, Eker S, et al. *All About Maude*. 4350 of *Lecture Notes in Computer Science*. Springer . 2007.

22. Romero JR, Rivera JE, Durán F, Vallecillo A. Formal and Tool Support for Model Driven Engineering with Maude. *Journal of Object Technology* 2007; 6(9): 187–207.

23. Rivera JE, Durán F, Vallecillo A. Formal Specification and Analysis of Domain Specific Models Using Maude. *Simulation* 2009; 85(11-12): 778–792.

24. Durán F, Moreno-Delgado A, Álvarez-Palomo JM. Statistical Model Checking of e-Motions Domain-Specific Modeling Languages. In: Stevens P, Wasowski A., eds. *19th Intl. Conf. Fundamental Approaches to Software Engineering (FASE)*. 9633 of *Lecture Notes in Computer Science*. ETAPS. Springer; 2016: 305–322.

25. de Oliveira PA, Moreno-Delgado A, Durán F, Pimentel E. Towards the predictive analysis of cloud systems with e-Motions. In: Pastor O, Guizzardi RSS, Kaplan GN, et al., eds. *XX Iberoamerican Conference on Software Engineering (CIbSE)*PUCPR. Curran Associates; 2017: 1–15.

26. de Oliveira PA, Durán F, Pimentel E. Towards the Performance Analysis of Elastic Systems with e-Motions. In: Cerone A, Roveri M., eds. *Software Engineering and Formal Methods*Springer; 2018: 475–490.

27. OMG*Business Process Model and Notation (BPMN) – Version 2.0*. . December 2011.

28. Becker S, Koziolek H, Reussner R. The Palladio component model for model-driven performance prediction. *J. of Systems and Software* 2009; 82(1): 3 - 22.

29. Troya J, Vallecillo A, Durán F, Zschaler S. Model-driven performance analysis of rule-based domain specific visual models. *Information & Software Technology* 2013; 55(1): 88–110.

30. Sinreich D. An architectural blueprint for autonomic computing. tech. rep., IBM; 2005.

31. Arcelli D. Exploiting Queuing Networks to Model and Assess the Performance of Self-Adaptive Software Systems: A Survey. *Procedia Computer Science* 2020; 170: 498 – 505. doi: 10.1016/j.procs.2020.03.108

32. Mian NA, Ahmad F. Modeling and Analysis of MAPE-K loop in Self Adaptive Systems using Petri Nets. *Int. J. Comput. Sci. Netw. Secur.* 2017; 17(12): 158–163.

33. Weyns D, Iftikhar U. Model-based simulation at runtime for self-adaptive systems. In: Kounev S, Giese H, Liu J., eds. *International Conference on Autonomic Computing (ICAC)*IEEE; 2016: 1–9.

34. Monshizadeh Naeen H, Zeinali E, Toroghi Haghighat A. Adaptive Markov-based approach for dynamic virtual machine consolidation in cloud data centers with quality-of-service constraints. *Software: Practice and Experience* 2020; 50(2): 161-183. doi: 10.1002/spe.2764

35. Bucchiarone A, Ehrig H, Ermel C, Pelliccione P, Runge O. Rule-based modeling and static analysis of self-adaptive systems by graph transformation. In: De Nicola R, Hennicker R., eds. *Software, services, and systems*Springer. 2015 (pp. 582–601).

36. Becker S, Brataas G, Lehrig S. *Engineering Scalable, Elastic, and Cost-Efficient Cloud Computing Applications*. Springer . 2017.

37. Krach SD, Scheerer M. SimuLizar NG: An extensible event-oriented simulation engine for self-adaptive software architectures. In: Eichelberger H, Schmid K., eds. *9th Symposium on Software Performance (SSP)*; 2018.

38. Huber N, van Hoorn A, Koziolek A, Brosig F, Kounev S. S/T/A: Meta-modeling run-time adaptation in component-based system architectures. In: Chao KM., ed. *9th Intl. Conf. on e-Business Engineering (ICEBE)*IEEE. ; 2012: 70–77.

39. Heinrich R. Architectural Run-time Models for Performance and Privacy Analysis in Dynamic Cloud Applications. *SIGMETRICS Perform. Eval. Rev.* 2016; 43(4): 13–22. doi: 10.1145/2897356.2897359

40. 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*Springer. 2009 (pp. 201–222).

41. Falkner K, Szabo C, Chiprianov V. Model-driven performance prediction of systems of systems. In: Kienzle J, Pretschner A., eds. *19th Intl. Conf. on Model Driven Engineering Languages and Systems*ACM/IEEE. ; 2016: 44–44.

42. Abuseta Y, Swesi K. Towards a framework for testing and simulating self adaptive systems. In: Surendra Prasad Babu M, Li W., eds. *6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*; 2015: 70–76.

43. Johnsen EB, Lin J, Yu IC. Comparing AWS Deployments Using Model-Based Predictions. In: Margaria T, Steffen B., eds. *7th Intl. Symp. on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA)*. 9953 of *Lecture Notes in Computer Science*. Springer; 2016: 482–496.

44. Bernardi S, Marrone S, Merseguer J, Nardone R, Vittorini V. Towards a model-driven engineering approach for the assessment of non-functional properties using multi-formalism. *Software & Systems Modeling* 2018: 1–24.