Solving the Next Adaptation Problem with Prometheus

Konstantinos Angelopoulos, Fatma Başak Aydemir, Paolo Giorgini, John Mylopoulos
University of Trento
Trento, Italy
email: {k.angelopoulos, fatmabasak.aydemir, paolo.giorgini, john.mylopoulos}@unitn.it

***Abstract*—Dealing with multiple requirement failures is an essential capability for self-adaptive software systems. This capability becomes more challenging in the presence of conflicting goals. This paper is concerned with the *next adaptation problem*: the problem of finding the best next adaptation in the presence of multiple failures. 'Best' here means that the adaptation chosen optimizes a given set of objective functions, such as the cost of adaptation or the degree of failure for system requirements. The paper proposes a formal framework for defining the next adaptation problem, assuming that we can specify quantitatively the constraints that hold between indicators that measure the degree of failure of each requirement and control parameters. These constraints, along with one or several objective functions, are translated into a constrained multi-objective optimization problem that can be solved by using an OMT/SMT (Optimization Modulo Theories/Satisfiability Modulo Theories) solver, such as OptiMathSAT. The proposed framework is illustrated with the Meeting Scheduler exemplar and a second, e-shop case study.**

I. Introduction

Self-adaptive software systems operate in uncertain environments and adapt their behaviour when their requirements are failing. Adaptation is accomplished through one or more feedback (aka Monitor-Analyze-Plan-Execute, or MAPE) loops that support monitoring the performance of the system and the environment (M), determining the root cause(s) of failures (A), selecting an adaptation (P), and carrying out the adaptation (E).

This paper focuses on the *next adaptation problem*, concerned with the selection of a new adaptation to address one or more failures. One of the main challenges for any adaptation mechanism is to select an optimal adaptation relative to one or more objective functions, such as minimizing cost of adaptation, minimizing degree of failure, and/or maximizing customer value. The paper advances our previous work [1] by proposing a new framework named Prometheus[1] that does not just choose a good adaptation for the failing requirements, but actually selects an optimal one, relative to user-specified objective functions. In particular, given an analytical model that describes the relation between requirements success rates and control parameters, and given a set of failing requirements, the adaptation mechanism searches for new values for control parameters that reduce the degree of failure, while optimizing given objective functions. If there are several objective functions, the adaptation chosen optimizes them lexicographically [2], i.e. best adaptations are selected relative to the most important objective function, among those best adaptations are selected relative to the second most important objective function, etc.

[1]In Greek mythology, Prometheus is the archetypical controlling personality, setting goals and making sure they get fulfilled.

The rest of this paper is structured as follows. Section II presents the research baseline for this work. Section III describes how finding the optimal next adaptation can be formulated as a constrained multi-objective optimization problem. Section IV introduces the *Prometheus* adaptation framework. Fi In Section V we compare *Prometheus* with $Zanshin$. Finally, in Section VI we contrast our proposal with related work, while in Section VII we conclude.

II. Research Baseline

This section presents the research baseline of our paper.

Goal Models. In continuation of our previous work [1] we use goal models for representing stakeholder requirements. A goal model captures both functional and non-functional requirements, referred as *hard goals* and *soft goals* respectively. Hard goals are *AND/OR*-refined until each goal is operationalized by tasks. Along with goals, *domain assumptions* represent preconditions that must hold for the system to operate properly. For example, in Fig. 1, the goal *Collect Automatically* cannot be satisfied unless the domain assumption *Participants use the system calendar* holds. Soft goals represent desired qualities of the system-to-be. In spite of their qualitative nature, soft goals can be operationalized by *quality constraints* that quantify the degree to which they are fulfilled. For example, *Fast Scheduling* may be operationalized by the quality constraint duration(Schedule Meeting) $\leq$ 6hrs. Alternatively, quality goals can be operationalized by optimization constraints, named *quality attributes*. For instance, *Fast Scheduling* may be operationalized by minimizing the time it takes to schedule meetings.

A goal model for a self-adaptive system captures all functional requirements for the system-to-be. The system at runtime can switch among alternative refinements where this is possible, in order to guarantee the satisfaction of all root goals. However, choices among alternatives can be constrained. For instance, as Fig. 2 shows, if the timetables are collected automatically then the meeting's date must be selected automatically by the system as well. Such relationships are called *goal constraints* and capture dependencies among goals.

Monitoring requirements is a crucial task for adaptation mechanisms. *Awareness Requirements* (*AwReqs*) [3] impose

978-1-4799-8710-8/16/$31.00 ©2016 IEEE

constraints over the success/failure of other requirements (including other *AwReqs*). When an *AwReq* is violated a new adaptation is triggered. For instance, if the goal *Collect Timetables* is not fulfilled more than twice a week then $AR1$ and $AR4$ fail. At runtime each *AwReq* is associated with a variable named *indicator* which measures the degree of success of the monitored requirement.

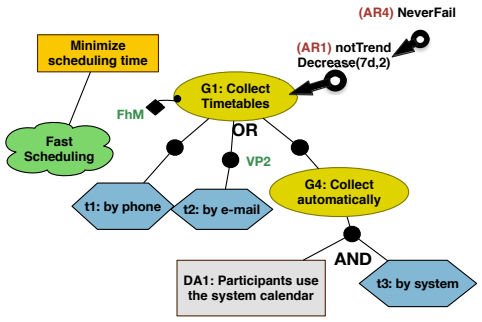

Figure 1. Meeting-Scheduler goal model fragment

Software Adaptation. The mandate of a self-adaptive system is to operate in uncertain environments and when one or more of its requirements fail, switch to an alternative configuration in order to recover. The set of all alternative configurations is referred as *adaptation space*. More specifically, an adaptation space is composed of a set of *control parameters* that influence the values of indicators. Control parameters are divided in *Variation Points* (VPs) and *Control Variables* (CVs) [4]. The first, is related to the explicit *OR*-refinements of the goal model. For instance, $VP3$ in Fig. 2 is assigned with values within the range $\{t7, t8\}$ depending on which of the two tasks is chosen for satisfying $G6$. On the other hand, CVs specify the amount of a resource that is used for the fulfilment of a goal. For example, the CV from how many participants (FhM) indicates from what percentage of the required participants to a meeting does one need to get timetables in order to fulfil the goal *Collect Timetables*. Clearly, this is a control variable because the lower its value, the easier it will be to fulfil the goal *Collect Timetables*. Changing the values of control parameter can result in change of success rates of certain indicators and quality attributes. For example, when FhM is increased $I3$ decreases whereas $find\_date\_time$ increases.

The feasibility of satisfying all goals varies as the environment changes (e.g. system's workload increases). In cases where a requirement R constantly fails to be fulfilled it is substituted by a new one R'. For example, if $(AR3)NeverFail$ depicted in Fig. 2 fails for more than two days, then it is substituted by a relaxed requirement $(AR3')SuccesssRate(90\%)$. The latter is referred as *Evolution Requirement* (*EvoReq*) [5] These requirements apply when certain preconditions, provided by the stakeholders, are satisfied.

Optimization Modulo Theories. The satisfiability modulo theories (SMT) problem is the problem of deciding the satisfiability of a quantifier-free first-order logic formula with respect to some decidable theory. When the theory includes linear arithmetic over the rationals (LRA), the first order logic formula whose satisfiability is questioned consists of atomic propositions and linear arithmetic constraints.

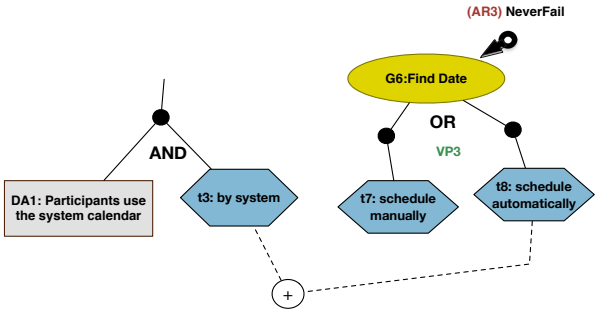

Figure 2. Goal constraint

Optimization Modulo Theories over LRA is the problem of finding solution(s) to an SMT(LRA) formula which optimize one or more rational-valued objective functions. There are very efficient SMT(LRA) and OMT(LRA) solvers, which combine the power of contemporary SAT solvers with dedicated linear-programming decision and minimization procedures. For instance, the solver OptiMathSAT [6] was able to handle problems with thousands of Boolean/rational variables in less than 10 minutes each.

In our work the logic formulas are the boolean refinements of the goal model. Therefore, the solver needs to find a set of tasks among all the alternatives in order for the root goal to be fulfilled. As opposed to traditional goal reasoning [7] approaches the selection is guided by a set of cost-functions that must be optimized as we will see in the next section.

III. The Next Adaptation Problem

In this section we describe how the problem of choosing values for control parameters when indicators fail can be formulated as a constrained multi-objective optimization problem. Then, we provide an instance of this problem based on the Meeting-Scheduler exemplar.

As explained in the previous section, tuning a control parameter results in a change of value of certain indicators and quality attributes. Therefore, after every diagnosis, the available goals and tasks are annotated with the potential contributions that can provide to the associated indicators as shown in Fig. 3. In this example the full form of the goal model (the possible values for CVs are represented as OR-refinements) is captured along with impact of each goal or task to indicators and quality attributes based on the *control parameter profile* presented in Table I. More specifically, increasing FhM from 70% to 100% will result in a decreased $I3$ by 1.5%, whereas increasing the percentage of maximum allowed conflicts (MCA) over the number of invitees from 20% to 40% increases $I3$ by 1.6%. Each alternative results in different amount of time that the goal *Find Date* requires to be

fulfilled and therefore there is an annotation with the value of the quality attribute $find\_date\_time$ for each of them. The values of Table I as well as those for the quality attributes are provided by domain experts and can be updated if necessary to increase precision.

Table I
CONTROL PARAMETER PROFILE.

ΔCP	$\Delta I3(\%)$
MCA	± 0.08
FhM	∓ 0.05
$VP3\{automatically \rightarrow manually\}$	$+6$

Each time one or more indicators fail, the goal model must be annotated based on what were the previous values of the control parameters and the control parameter profile.

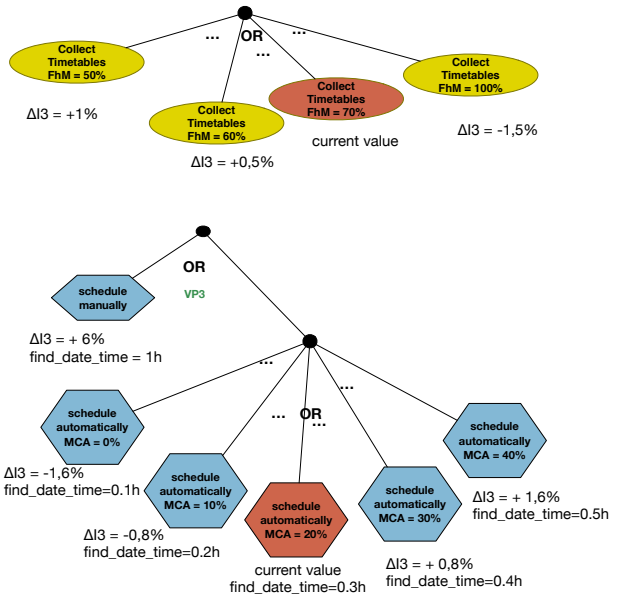

Figure 3. Goal model annotated with contributions

When the measured value of an indicator I_j^m is below the threshold I_j^o by $R_j = I_j^o - I_j^m$, a new adaptation is triggered in order to minimize R_j (ideally, it should be zero) for every indicator.

Definition 1 (Indicator Cost-Function): Let $D_I = \{I^m, I^o, R, AS'\}$, be the diagnosis for the indicator I, where I^m is its measured value, I^o is its threshold, $R = I^o - I^m$ and AS' the set of all available goals or tasks that can contribute to the current value of I positevely, negatively or zero. An Indicator Cost-Function F^I is defined as $F^I = R + \sum \Delta I$, where $\sum \Delta I$ is the sum of contributions that I will receive by the next adaptation.

According to Fig. 3 if the next adaptation includes $FhM = 60\%$ and *schedule manually* is selected, $\sum \Delta I3 = 0.5 + 6 = 6.5\%$. Therefore, the indicator $I3$ is going to be increased by 6.5%. The target of the adaptation mechanism is to minimize all Indicator Cost-Functions. However, due to the presence of conflicting contributions among the indicators the adaptation mechanism needs to settle for a trade-off. Towards this direction, we prioritize all indicators using Analytical Hierarchy Process (AHP) [8], [9], eliciting weights that represent their importance.

Definition 2 (Global Cost-Function): Let F be the set of all Indicator Cost-Functions and W the set of their respective weights. A Gobal Cost-Function F^G is defined as $F^G = \sum w_j \times F_j^I$, where $w_j \in W$ and $F_j^I \in F$.

The role of the adaptation mechanism is twofold. First, a configuration of the goal model must be found so that the root goal is satisfied while the Global Cost-Function is minimized. In other words, the next adaptation problem consists of a combination of two different problems a) satisfiability of all constraints and b) multi-objective optimization. Such combined problems are solved by reasoning technologies, notable Satisfiability and Optimization Modulo Theories (SMT/OMT) [6].

In Fig. 3, apart from the contributions to the indicators, goals and tasks are also annotated with certain kinds of costs. For example, the value selection for MCA includes the amount of time it takes to schedule a meeting's date (find_date_time), whereas the value for FhM determines the time it takes to collect timetables ($collection\_time = FhM \times 0.02$). Stakeholders, usually require the satisfaction of their goals with the minimal cost adaptation. This means that the total time for scheduling a meeting ($total\_scheduling\_time$ = $collection\_time$ + $find\_date\_time$) must also be minimized. The *Next Adaptation Problem* can encompass optimizations relative to other costs, such as $total\_scheduling\_time$.

Definition 3 (Next Adaptation Problem): Let $P = \{F^G, QA_1, ..., QA_n, AS'\}$ be a tupple where F^G is a global cost-function, $QA_1, ..., QA_n$ a set of quality attributes, and AS' the new adaptation space that includes all the available control parameters. The Next Adaptation Problem refers to finding an optimal configuration over the goal model that minimizes F^G and lexicographically optimizes each quality attribute (based on stakeholder preferences), wrt to the availability of goals and tasks in AS'.

The process followed to formulate the Next Adaptation Problem is the following. First, all *AwReqs* and control parameters must be elicited, as well as the control parameter profile that captures the relationship between the two. For every *AwReq* there is a threshold I^o and an indicator I^m that measures the actual success rate at runtime. Then, the designers select an initial configuration over the goal model and as we will see in the next section the *Prometheus* framework automatically annotates each task of the goal model with the contributions ΔI to every indicator that influence in the same manner as in Fig.3. This process is elaborated with the use of CGM-tool [10], which is built on the top of OptiMathSAT and allows the definition of cost-functions the values of which depend on the configuration of the goal model. These cost-functions are provided by Definition 1 and 2 along with the quality attributes of the system.

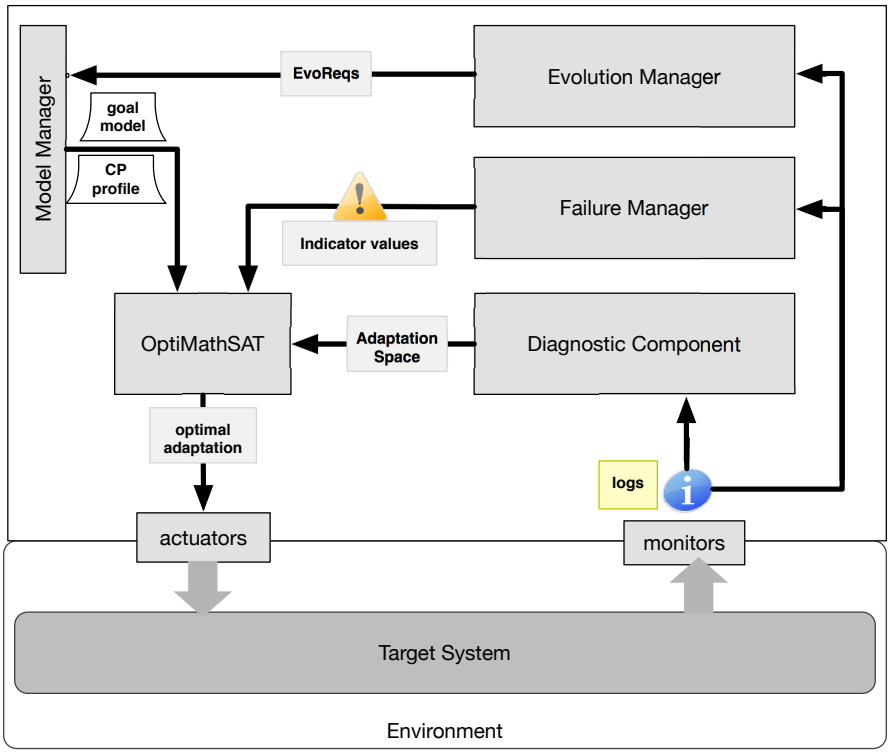

Figure 4. The Prometheus framework

IV. Prometheus Framework

In the previous section we presented how the adaptation process is modelled as an optimization problem using goal models and a quantitative information about the relationship between control parameters and indicators. This section describes the steps of the adaptation process at runtime as well as the proposed Prometheus framework, whose architecture is shown in Fig. 4.

Prometheus interacts with the target system and its environment through monitors and actuators that are responsibility of the system designers to build and usually are application-specific. The internal mechanism of *Prometheus* consists of five components described below.

Diagnostic Component This component reads the system logs and reasons about the root causes of the identified failures. More specifically, discovers denied domain assumptions or failing tasks that could not be performed. These domain assumptions and tasks are marked as *denied* in the new available adaptation space constraining the available options for finding the optimal next adaptation. For instance, if the domain assumption $DA1$ in Fig. 2 is denied then $t3$ cannot be selected as an option for collecting timetables and due to a goal constrain neither $t8$ can be selected for finding a date. Therefore, the diagnostic tool is responsible for eliminating all the solutions that are unfeasible due to unsatisfied preconditions, component failures or unavailable resources.

Failure Manager This component reads the system logs and measures the success rates of each indicator. When the measured value of one or more indicators is found below the threshold imposed by the associated *AwReq* a new adaptation is triggered and a new configuration over the goal model must be chosen.

Evolution Manager This component reads system logs and checks if any precondition holds, if it does, the goal model is updated in accordance with the *EvoReq*.

Model Manager This component stores the control parameter profile of the system and the elicited goal model. Each time a new adaptation is triggered the goal model is annotated with the impact values of each goal to the indicators.

OptiMathSAT This component receives the annotated goal model along with the new adaptation space that excludes a certain amount of alternative configurations. Based on this model and the measured values of the indicators produces an optimal next adaptation.

As we mentioned earlier our work provides an implementation of the MAPE loop that has been proposed as a reference framework in order to engineer self-adaptive systems. *Prometheus* is an implementation of this framework where the Failure Manager along with the Diagnostic Component perform the Analyze task whereas OptiMathSAT performs the Plan task.

To give a better understanding of how the framework operates, we describe every step followed for finding the next

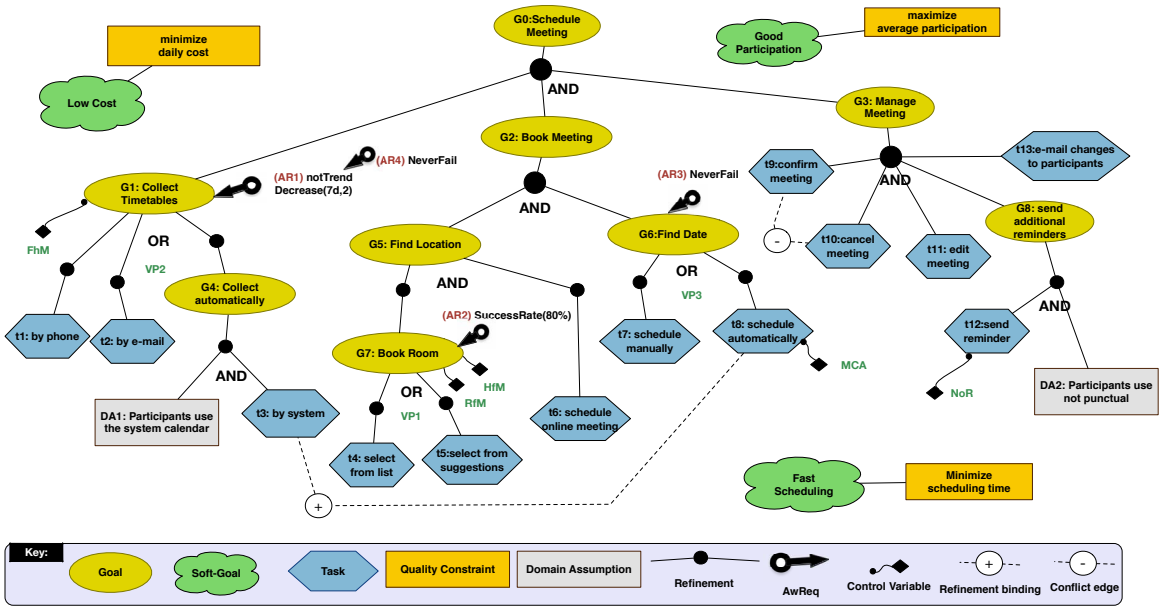

Figure 5. Meeting-Scheduler goal model

adaptation. The steps are:

1) **Collect system logs.** The success or failure of the executed tasks is recorded in logs collected by the designed monitors.
2) **Detect failures.** The Failure Manager compares measured values of the indicators with those that are imposed by their associated *AwReqs*. If one or more failures are detected a new adaptation is triggered.
3) **Find root causes.** The diagnostic tool provides the new adaptation space which excludes the goals that caused the detected failures and marks as denied the domain assumptions that do not hold any more.
4) **Apply *EvoReqs*.** The Evolution Manager updates the goal model with *EvoReqs* if any of the preconditions specified applies.
5) **Annotate the goal model.** The Model Manager annotates the goal model based on the control parameter profile.
6) **Find Optimal Next Adaptation.** OptiMathSAT produces an optimal adaptation.
7) **Apply new adaptation.** The new adaptation is applied to the target system by the actuators.

V. Evaluation

In this section we describe the Meeting-Scheduler and e-shop exemplars as well as the evaluation process of *Prometheus* by using failure scenarios.

A. The Meeting-Scheduler Exemplar

The main goal of the Meeting-Scheduler application is to receive meeting requests and produce meeting bookings. Fig. 5 captures system goals. For the top goal to be satisfied, timetables must be collected (satisfy $G1$), make a booking for every meeting (satisfy $G2$) and allow the meeting organizers to manage their meetings (satisfy $G3$). The timetables can be collected by phone, by e-mail, or automatically by the system. However, the third option is available only if the meeting participants are using the system calendar. Next, booking a meeting involves finding a location and an appropriate date. The system offers at the same time the opportunity to book a room or schedule an online meeting in case rooms are not available. A room can be selected from the entire list of existing rooms or from the suggested rooms by the system ($t4$ and $t5$ respectively). In the same manner, a date can be selected manually by the meeting organizer or the system selects one automatically ($t7$ and $t8$ respectively). A date though can be selected automatically only if the timetables are collected automatically and vice versa. Finally, meeting organizers can confirm, cancel and edit their meetings ($t9$, $t10$ and $t11$ respectively), while the system is responsible for notifying the participants in case a modification takes place ($t13$) and send more reminders in case the participants are not punctual enough.

To monitor the success of these requirements, four *AwReqs* are placed over certain goals that are prone to failure. $AR1$ dictates that goal $G1$ must not fail more than twice a week and $AR4$ imposes on $AR1$ to never fail. Next, $AR2$ prescribes that at least for 80% of the meetings a room must have successfully be found and $AR3$ that $G6$ must always be fulfilled.

In addition to FhM and MCA we presented in the previous section, other control parameters are available to regulate the success rate of the indicators associated to the aforementioned *AwReqs*. First, the goal *Book Room* is related to two CVs that control the number of local rooms (RfM) and hotel rooms (HfM) reserved for meetings. The number of additional

Table II
CONTROL PARAMETER PROFILE.

ΔCP	$\Delta I2(\%)$	$\Delta I3(\%)$	$\Delta I4(\%)$
MCA	0	± 0.08	0
FhM	0	∓ 0.05	0
RfM	± 2.1	0	0
HfM	± 2.7	0	0
NoR	0	0	0
$VP1\{t4 \rightarrow t5\}$	$+2$	0	0
$VP1\{t5 \rightarrow t4\}$	-2	0	0
$VP2\{t1 \rightarrow t2\}$	0	0	-2
$VP2\{t1 \rightarrow G4\}$	0	0	$+6$
$VP2\{t2 \rightarrow t1\}$	0	0	$+4$
$VP2\{t2 \rightarrow G4\}$	0	0	$+6$
$VP2\{G4 \rightarrow t1\}$(DA1 is true)	0	0	-5
$VP2\{G4 \rightarrow t2\}$(DA1 is true)	0	0	-6
$VP2\{G4 \rightarrow t1\}$(DA1 is false)	0	0	$+5$
$VP2\{G4 \rightarrow t2\}$(DA1 is false)	0	0	$+3$
$VP3\{t7 \rightarrow t8\}$	0	-2	0
$VP3\{t8 \rightarrow t7\}$	0	$+6$	0

reminders NoR associated to task $t12$ is yet another CV. Along with the CVs there are three VPs that stem from the OR-refinements of the goal model. In Table II the full control parameter profile for the Meeting-Scheduler application is presented.

```
! I2 = 72%, I3 = 94%, I4 = 87%
Current Configuration:
VP1=t5, VP2=t3, VP3=t8
MCA=10, FhM=70, NoR=0, RfM=6, HfM=4
! DA1 = false  DA2 = false
! No EvoReqs apply
P1: MCA=20, FhM=78, NoR=0, RfM=12, HfM=3
        VP1=t4, VP2=t1, VP3=t7
P2: MCA=20, FhM=78, NoR=0, RfM=3, HfM=10
        VP1=t5, VP2=t1, VP3=t7
_____________________________________

Output
P1: I2=79.9%, I3=100%, I4=92%
        total_cost = 90
        average_participation = 91.6%
        total_scheduling_time = 6.6hrs
P2: I2=78%, I3=100%, I4=92%
        total_cost = 230
        average_participation = 91.6%
        total_scheduling_time = 6.6hrs
```

Figure 6. *Prometheus* Output

In Fig. 5 three soft goals are specified to capture the non-functional requirements of the system. The *Low Cost* soft goal is associated with the quality attribute $daily\_cost$. The cost of a hotel rooms is 20€ and the daily cost for calls is $call\_cost = 30$€ if timetables are collected by phone, otherwise $call\_cost = 0$€. Therefore $daily\_cost = 20 \times HfM + call\_cost$ is a quality attribute that must be minimized. Next, the soft goal *High Participation* is associated with the quality attribute $average\_participation = 80 + 0.2 \times FhM - 0.2 \times MCA + 5 \times NoR(\%)$ which must be maximized. When $VP3 = t7$ then $MCA = 20$. Finally, the soft goal *Fast Scheduling* is associated with the quality attribute $total\_scheduling\_time$ which calculated as described in Section IV.

```
! I2 = 72%, I3 = 94%, I4 = 87%
Current Configuration:
VP1=t5, VP2=t3, VP3=t8
MCA=10, FhM=70, NoR=0, RfM=6, HfM=4
! DA1 = false  DA2 = false
! No EvoReqs apply

_____________________________________

Output
Iteration 1
Z:  I2=72.54%, I3=100%, I4=92%
Z:  MCA= 20, FhM=70, NoR=0, RfM=6, HfM=6
    VP1=t5, VP2=t1, VP3=t7
    total_cost = 150
        average_participation = 90%
        total_scheduling_time = 6.6hrs
Iteration 2
Z:  I2=73.58%, I3=100%, I4=92%
Z:  MCA= 20, FhM=70, NoR=0, RfM=6, HfM=8
    VP1=t5, VP2=t1, VP3=t7
    total_cost = 190
        average_participation = 90%
        total_scheduling_time = 6.6hrs
Iteration 3
Z:  I2=74.46%, I3=100%, I4=92%
Z:  MCA= 20, FhM=70, NoR=0, RfM=10, HfM=8
    VP1=t5, VP2=t1, VP3=t7
    total_cost = 190
        average_participation = 90%
        total_scheduling_time = 6.6hrs
Iteration 4
Z:  I2=75.84%, I3=100%, I4=92%
Z:  MCA= 20, FhM=70, NoR=0, RfM=14, HfM=8
    VP1=t5, VP2=t1, VP3=t7
    total_cost = 190
        average_participation = 90%
        total_scheduling_time = 6.6hrs
Iteration 5
Z:  I2=79.44%, I3=100%, I4=92%
Z:  MCA= 20, FhM=70, NoR=0, RfM=18, HfM=10
    VP1=t5, VP2=t1, VP3=t7
    total_cost = 230
        average_participation = 90%
        total_scheduling_time = 6.6hrs
Iteration 6
Z:  I2=82.5%, I3=100%, I4=92%
Z:  MCA= 20, FhM=70, NoR=0, RfM=22, HfM=10
    VP1=t5, VP2=t1, VP3=t7
    total_cost = 230
        average_participation = 90%
        total_scheduling_time = 6.6hrs
```

Figure 7. $Zanshin$ Output

To evaluate our approach we implemented a simulation of the Meeting-Scheduler application. In this simulation we encoded a failure scenario and inserted it as input to both*Prometheus* and $Zanshin$ integrated with the $Qualia$ as described in [11]. Both frameworks are requirement-based and this has been our main motivation for carrying out this comparison. However, $Zanshin$ uses qualitative information

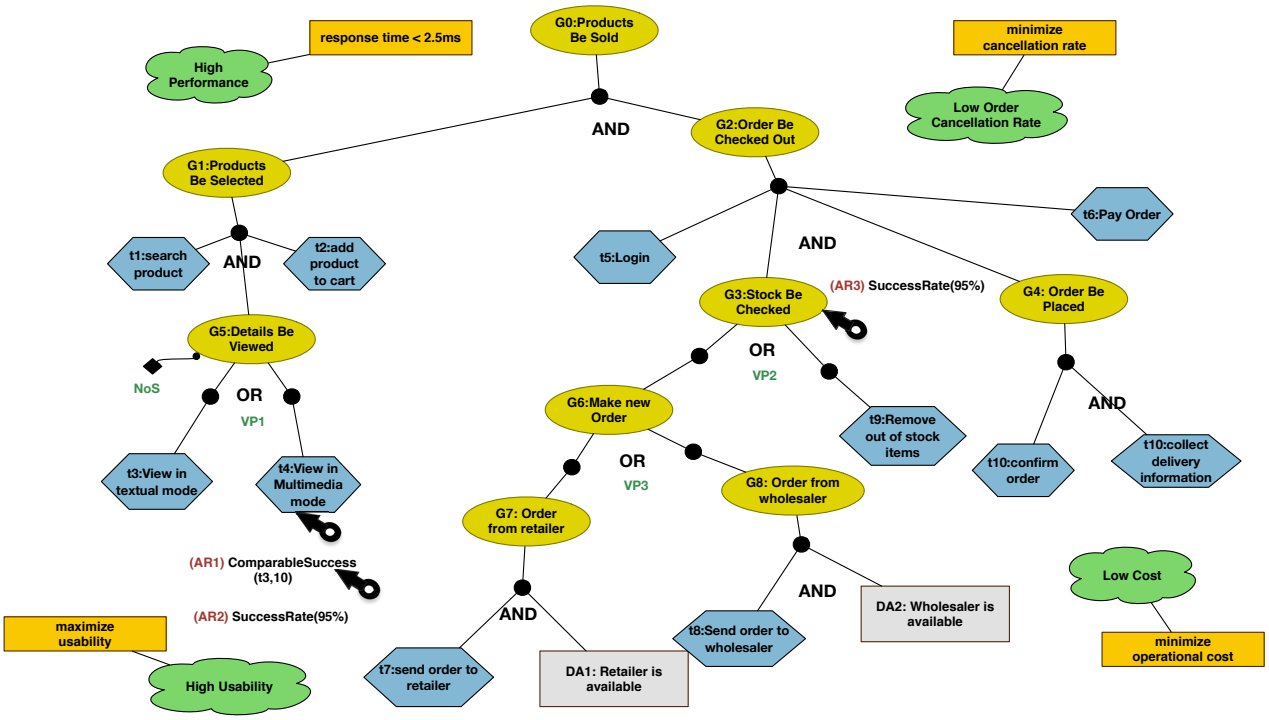

Figure 8. E-shop goal model

for capturing the impact of control parameters on indicators and does not optimize as opposed to the quantitative and optimizing approach of *Prometheus*. Moreover, the default adaptation algorithm of $Zanshin$ selects randomly a control parameter that contributes positively to the treated failure, which is increased by a predefined amount of units. In order also to demonstrate the importance of lexicographic optimization we execute the adaptation process of *Prometheus* with and without optimizing quality attributes. Among the quality attributes cost is optimized first, participation second and scheduling time third. In Fig. 6 the results of the simulation are presented. $P2$ marks the next adaptation and the consequent output of *Prometheus* when only the Global Cost-Function is optimized whereas $P1$ also optimizes (lexicographically) with respect to quality attributes. Finally, Z in Fig. 7 indicates the next adaptation and output produced by $Zanshin$.

The results in Fig. 6 and Fig. 7 show that *Prometheus* performs better than $Zanshin$ since it exploits quantitative relations between control parameters and indicators. On the other hand, $Zanshin$ changes by a fixed amount randomly chosen control parameters that is known to improve the failing indicators [12]. This means that it would take more iterations for $Zanshin$ to fix a failure or to minimize it. More specifically, in the simulated failure scenario it took six steps until $I3$ and $I4$ converges to the best possible value whereas $I2$ slightly overshoots. Moreover, the results indicate that the qualities of the system are not taken into account while deciding the next adaptation. The large number of iterations for the indicators to converge to their prescribed thresholds, implies that $Zanshin$ is not suitable for rapidly changing environment, since by the tame a good solution is found the failing indicators might be different. Given the result of the simulation, lexicographic optimization can provide the same results for the failing indicators as the optimization of the Global Cost-Function only, but also considerably improves quality attributes, such as $total\_cost$ in this case.

B. The E-shop Exemplar

The main goal of the e-shop application is to place orders of goods that clients buy online. Fig. 8 captures the goals for this system. For the top goal to be satisfied the customer must select the product they would like to order (goal $G1$) and check-out the order (goal $G2$). The customer is able to search for products they are interested in (task $t1$), view the details of the product (goal $G5$) in textual mode (task $t3$) or multimedia mode (task $t4$). For an order to be checked out the customer must fist login (task $t5$). Then, the product's availability is checked (goal $G3$). The goal $G3$ can be fulfilled either by making a new order (goal $G6$) or by removing the selected item from the stock list (task $t9$). The product is ordered either from a retailer (goal $G7$) or from a wholesaler (goal $G8$). A precondition for sending an order to a retailer or a wholesaler (tasks $t7$ and $t8$ respectively) requires a retailer or a wholesaler to be available ($DA1$ and $DA2$ respectively).

For the requirements of this exemplar we elicited three *AwReqs*. $AR1$ prescribes that multimedia mode for viewing product details must by used 10 times more than textual model

and according to $AR2$ this constraint must not fail more than 80% of the times. Next, the goal $G2$ must not fail more than 95%.

The elicited *AwReqs* are related to three control parameters. $AR2$ can be controlled by changing the value of $VP1$, or in other words switching textual to multimedia mode and vice versa and the number of servers (NoS) that are deployed to host the webpage of the e-shop. Finally, $VP1$ and $VP2$ can be used to control the success of $AR3$.

Table III
CONTROL PARAMETER PROFILE.

ΔCP	$\Delta I2(\%)$	$\Delta I3(\%)$	response time(ms)	cancellation rate(%)
NoS	± 1.2	0	∓ 200	0
$VP1\{t3 \rightarrow t4\}$	3	0	$+1000$	0
$VP1\{t4 \rightarrow t4\}$	3	0	$+200$	0
$VP1\{t4 \rightarrow t3\}$	-1	0	$+1$	0
$VP1\{t3 \rightarrow t3\}$	-0.2	0	0	0
$VP2\{G6 \rightarrow t9\}$	0	$+3.4$	0	$+4$
$VP2\{t9 \rightarrow G6\}$	0	0	0	-4
$VP3\{G7 \rightarrow G8\}$ (DA1 is false)	0	$+1.2$	0	0
$VP3\{G8 \rightarrow G7\}$ (DA2 is false)	0	$+0.8$	0	0

In Fig. 8 four soft goals are specified to capture non-functional requirements for the e-shop exemplar. First, *High Performance* is associated with the quality attribute $response\_time$ of the deployed servers, which in this case is not only minimized but also constrained to be lower than 2.5s. The disposal of more servers results in lower response time. Next, *High Usability* is associated to the quality attribute $usability$ which takes the value 5 when the website is viewed in multimedia mode and the value 3 when text mode is selected. Next, *Low Cost* relates to the quality attribute $operational\_cost = server\_cost + ordering\_cost$, where $server\_cos = 120 \times NoS$ and $ordering\_cost$ is the cost of making a new order which is 1200€ in case the products are ordered from a wholesaler and 1000€ in case it ordered from a retailer. The prices might vary through time since they depend on which products are mostly sold in a particular period of time and in what quantities. Therefore, it is responsibility of the domain experts to update this numbers. Finally, *Low Order Cancellation Rate* is associated with the quality attribute $cancellation\_rate$.

As for the Meeting-Scheduler exemplar also in this case we simulated a failure scenario and we compare the responses of *Prometheus* and $Zanshin$. The control parameter profile is presented in Table III whereas the outputs of *Prometheus* and $Zanshin$ are depicted in Fig. 9 and Fig. 10 respectively.

The results in Fig. 9 and Fig. 10 show that both frameworks managed to find good solutions for the failing indicators. However, as in the previous case *Prometheus* managed to find optimal solutions for the soft goals as well.

C. Discussion

The experiments ran on a computer with an Intel i5 processor at 2.5GHz and 16GB of RAM. Both $OptiMathSAT$

```
! I2 = 93%, I3 = 94%
!response time = 4000ms
!cancellation rate = 10 %
Current Configuration:
VP1=t4, VP2=G6, VP3=G7
NoS=3
! DA1 = false  DA2 = true
! No EvoReqs apply

P:  NoS=4
    VP1=t4, VP2=G6, VP3=t8
  ---------------------------------
Output
P:  I2=99.6%, I3=95.2%
        usability = 5
        response time = 1400ms
        cancellationRate =6 %
        cost = 1920 euro
```

Figure 9. *Prometheus* Output

```
! I2 = 93%, I3 = 94%
!response time = 4000ms
!cancellation rate = 10 %
Current Configuration:
VP1=t4, VP2=G6, VP3=G7
NoS=3
! DA1 = false  DA2 = true
! No EvoReqs apply
  ---------------------------------
Output
Iteration 1
Z:  NoS=4
        VP1=t4, VP2=t9, VP3= -
Z:  I2=97.2%, I3=97.4%
        usability = 5
        response time = 4000ms
        cancellationRate = 14 %
        cost = 480 euro
```

Figure 10. $Zanshin$ Output

[6] and its extension $CGM - tool$ [10] which has been the basis of the *Prometheus* prototype have been tested in terms of scalability and can handle up to 8000 nodes in negligible time. Compared to $Zanshin$, *Prometheus* is able to select among all the equivalent solutions the one that optimizes the quality attributes of the system as well.

The main bottleneck of the approach is deriving the control parameter profile, which is a human-driven process and the time overhead depends on the amount of the control parameters and the level of expertise of the software designers. However, related work [13] can be integrated with our approach in order to correct at runtime the values of the control parameter profile.

Another aspect to consider is the human intervention in the decision-making process. In safety critical systems there are unpredictable occasions where human decision must override those of the adaptation mechanism. Such systems are out of the scope of this work. However, exploring the human factor

in the adaptation process and the preconditions that must be satisfied for overriding the automated decisions is part of our future work.

VI. Related Work

In the field of requirements-driven adaptation many approaches that use goal models as a guide for deciding the next adaptation have been proposed. First, Wang et al. in [14] propose the use of the same diagnostic tool [15] as in our work to identify failing tasks and and select an alternative configuration over the elicited goal model based on qualitative contributions from hard goals to soft goals. This work, similarly to ours, can handle dynamic adaptation spaces, but qualitative reasoning cannot provide precision nor does it support optimization, as *Prometheus* does.

In [16] the authors propose the use of a cost-function for optimizing the non-functional requirements of the target system while minimizing the number of penalties taken for violating Service Level Agreements. Compared to our work the emphasis of this approach is given exclusively to non-functional requirements whereas in our work we examine failures mainly for functional one and find the next adaptation that minimizes/maximizes any additional quality attribute of the system through lexicographic optimization. Both approaches elicit control parameters from goal models. However, our approach selects which of them must be tuned based on the control parameter profile that describes the quantitative impact on the indicators and the availability in the adaptation space. On the other hand, the compared approach prioritizes the available control parameters and the selection is based on qualitative information about the impact on the goals.

Another goal-based approach is described in [13]. Here, goal reasoning techniques [7] are applied for selecting the best alternative over the goal model with respect to the level of satisfaction of the quality attributes that are present. The authors make the assumption that the preferences based on which the goal reasoning is performed might be inaccurate or might vary as the environments keeps changing. Their proposed solution to this problem is to apply a PID controller for finding the right contributions of each alternative to the monitored goals. This approach can be complementary to our work for correcting inaccuracies of in the control parameter profile.

On the side of architecture-based approaches, Rainbow [17] has an adaptation space of predefined strategies and when one or more quality attributes, such as response time and operational cost, are not satisfied their imposed thresholds, the strategy that maximizes the overall utility is selected. In the same research line, in [18] the authors propose the use of probabilistic model checking techniques to compose dynamically adaptation strategies taking also into account latencies about when the impact of a change in a control parameter will appear to the system's output. Our approach does not take into account such latencies. However, we take into consideration that the adaptation space is dynamic whereas Rainbow and its extension make the assumption that each control parameter has always the same impact on the system's goals.

Recent work [19], proposes an automated solution by applying formal control theory in order to derive an adaptation. The solution is based on on a simple and qualitative dynamic model which is identified online. However, the proposed solution in [19] works only for systems where a single goal is related to a single control parameter, and therefore cannot handle multiple failures. An extension of this work [20] deals with the presence of multiple objectives but it is limited to the fact that each of them can be controlled by a single control parameter. Moreover, as dependencies among control parameters are not taken into account as in our work with the use of goal relationships.

VII. Conclusions

We have proposed a framework that can compose an optimal adaptation when one or more requirements fail. The optimal nature of the produced adaptation refers to the minimization of the degree of failure of the system's functional requirements while non-functional properties of the system are optimized lexicographically according to stakeholder priorities.

Our proposed framework is built on top of a diagnostic component that reads the logs of the monitored system and reasons about the causes of failure. Failing domain assumptions and tasks define the new adaptation space where all the potential solutions lie in. Then, each alternative of the adaptation space is annotated with the quantitative impact that will bare to the indicators. Finally, the *OptiMathSAT* solver finds the best alternative in the new adaptation space.

The contribution of this work, is a requirements-driven approach that determines optimal adaptations for multiple failures and with respect to multiple objective functions. Moreover, we have demonstrated experimentally that our proposal performs better than a qualitative, requirements-driven framework ($Zanshin$), and also established that our proposal works for real world-size problems.

Of course, our proposal needs further evaluation with larger case studies to determine how usable it is by designers of adaptive software systems.

Acknowledgment

This work has been supported by the ERC advanced grant 267856 Lucretius: "Foundations for Software Evolution" (April 2011 - March 2016, http://www.lucretius.eu).

References

[1] K. Angelopoulos, V. E. S. Souza, and J. Mylopoulos, "Dealing with multiple failures in zanshin: a control-theoretic approach," in *9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2014, Proceedings, Hyderabad, India, June 2-3, 2014*, 2014, pp. 165–174. [Online]. Available: http://doi.acm.org/10.1145/2593929.2593936

[2] H. Isermann, "Linear lexicographic optimization," *Operations-Research-Spektrum*, vol. 4, no. 4, pp. 223–228, 1982.

[3] V. E. S. Souza, A. Lapouchnian, W. N. Robinson, and J. Mylopoulos, "Awareness requirements for adaptive systems," in *2011 ICSE Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2011, Waikiki, Honolulu , HI, USA, May 23-24, 2011*, 2011, pp. 60–69. [Online]. Available: http://doi.acm.org/10.1145/1988008.1988018

[4] V. E. S. Souza, A. Lapouchnian, and J. Mylopoulos, "System identification for adaptive software systems: A requirements engineering perspective," in *Conceptual Modeling - ER 2011, 30th International Conference, ER 2011, Brussels, Belgium, October 31 - November 3, 2011. Proceedings*, 2011, pp. 346–361. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-24606-7_26

[5] V. E. S. Souza, A. Lapouchnian, K. Angelopoulos, and J. Mylopoulos, "Requirements-driven software evolution," *Computer Science-Research and Development*, vol. 28, no. 4, pp. 311–329, 2013.

[6] R. Sebastiani and P. Trentin, "Optimathsat: A tool for optimization modulo theories," in *Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part I*, 2015, pp. 447–454. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-21690-4_27

[7] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani, "Reasoning with goal models," in *Conceptual Modeling—ER 2002*. Springer, 2003, pp. 167–181.

[8] J. Karlsson and K. Ryan, "A cost-value approach for prioritizing requirements," *Software, IEEE*, vol. 14, no. 5, pp. 67–74, Sep 1997.

[9] T. L. Saaty, "Desicion making by the analytic hierarchy process: Theory and applications how to make a decision: The analytic hierarchy process," *European Journal of Operational Research*, vol. 48, no. 1, pp. 9 – 26, 1990. [Online]. Available: http://www.sciencedirect.com/science/article/pii/037722179090057I

[10] C. M. Nguyen, R. Sebastiani, P. Giorgini, and J. Mylopoulos, "Multi object reasoning with constrained goal model," *arXiv preprint arXiv:1601.07409*, 2016.

[11] V. E. S. Souza, A. Lapouchnian, and J. Mylopoulos, *On the Move to Meaningful Internet Systems: OTM 2012: Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2012, Rome, Italy, September 10-14, 2012. Proceedings, Part I*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, ch. Requirements-Driven Qualitative Adaptation, pp. 342–361. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-33606-5_21

[12] ——, "Requirements-driven qualitative adaptation," in *On the Move to Meaningful Internet Systems: OTM 2012*. Springer, 2012, pp. 342–361.

[13] X. Peng, B. Chen, Y. Yu, and W. Zhao, "Self-tuning of software systems through dynamic quality tradeoff and value-based feedback control loop," *Journal of Systems and Software*, vol. 85, no. 12, pp. 2707–2719, 2012.

[14] Y. Wang and J. Mylopoulos, "Self-repair through reconfiguration: A requirements engineering approach," in *Automated Software Engineering, 2009. ASE '09. 24th IEEE/ACM International Conference on*, Nov 2009, pp. 257–268.

[15] Y. Wang, S. A. McIlraith, Y. Yu, and J. Mylopoulos, "An automated approach to monitoring and diagnosing requirements," in *Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '07. New York, NY, USA: ACM, 2007, pp. 293–302. [Online]. Available: http://doi.acm.org/10.1145/1321631.1321675

[16] P. Zoghi, M. Shtern, and M. Litoiu, "Designing search based adaptive systems: a quantitative approach," in *9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2014, Proceedings, Hyderabad, India, June 2-3, 2014*, 2014, pp. 7–16. [Online]. Available: http://doi.acm.org/10.1145/2593929.2593935

[17] S. Cheng, D. Garlan, and B. R. Schmerl, "Architecture-based self-adaptation in the presence of multiple objectives," in *Proceedings of the 2006 international workshop on Self-adaptation and self-managing systems, SEAMS 2006, Shanghai, China, May 21-22, 2006*, 2006, pp. 2–8. [Online]. Available: http://doi.acm.org/10.1145/1137677.1137679

[18] J. Cámara, D. Garlan, B. R. Schmerl, and A. Pandey, "Optimal planning for architecture-based self-adaptation via model checking of stochastic games," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, April 13-17, 2015*, 2015, pp. 428–435. [Online]. Available: http://doi.acm.org/10.1145/2695664.2695680

[19] A. Filieri, H. Hoffmann, and M. Maggio, "Automated design of self-adaptive software with control-theoretical formal guarantees," in *Proceedings of the 36th International Conference on Software Engineering*. ACM, 2014, pp. 299–310.

[20] ——, "Automated multi-objective control for self-adaptive software design," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE*, 2015, pp. 13–24.