A Goal-Oriented Approach for Adaptive SLA Monitoring: a Cloud Provider Case Study

Antoine Toueir, Julien Broisin, Michelle Sibilla

A Goal-Oriented Approach for Adaptive SLA Monitoring: a Cloud Provider Case Study

Antoine Toueir, Julien Broisin, Michelle Sibilla
IRIT, University Toulouse III - Paul Sabatier, 118 rue de Narbone, 31062 Toulouse, France
Email: {toueir,broisin,sibilla}@irit.fr

***Abstract*—We argue in this paper that autonomic systems need to make their integrated monitoring adaptive in order to improve their "comprehensive" Quality of Service (QoS). We propose to design this adaptation based on high level objectives (called *goals*) related to the management of both the "functional system QoS" and the "monitoring system QoS". Starting from some previous works suggesting a model-driven adaptable monitoring framework composed of 3 layers (configurability, adaptability, governability), we introduce a methodology to identify the functional and monitoring high level goals (according to the agreed Service Level Agreement - SLA) in order to drive models' instanciations. This proposal is first applied to a cloud provider case study for which two high level goals are developed (respect metrics freshness and minimize monitoring cost), and then simulated to show how the quality of management decisions, as well as intelligent monitoring of dynamic SLA, could be improved.**

I. Introduction

Autonomic systems that are implemented by virtue of their four characteristics self-configuration, self-optimization, self-healing, and self-protection, are serving the ultimate objective of making them self-managed to achieve high level objectives [1]. These objectives are strongly related to the QoS level provided by autonomic systems. When large and complex systems are targeted, the self-management characteristic (self-*) is a key issue to deal with. Basically, self-management is thought as the auto-adaptation capability that brings the system to reach an absolute or preferable state. Concretely, the four self-* characteristics are realized by implementing the MAPE-K (*Monitoring, Analyzing, Planning, Executing - Knowledge*) loop. This implementation is either integrated within an equipment, or into a system.

In this loop, the *Monitoring* system plays a crucial role, since wrong decisions might be taken by the *Analyzing* and *Planning* systems. Therefore, autonomic systems need to ensure the quality of this information (e.g. correctness, freshness, timeliness, accuracy, etc.). Moreover, within autonomic systems, monitoring is usually QoS-oriented. This QoS relies on the functional system (through the SLAs that have been agreeded with clients), but also on the management system; since the QoS specifications could be renegotiated or modified afterward, the monitoring system has to adapt its behavior according to these new requirements and constraints. In other words, monitoring systems have to be capable of (re)configuring their mechanisms based on quality requirements.

Our proposal deals with questions such as: how should the autonomic system guarantee the agreed QoS if a service provider hosts new services? What should the autonomic system do if a part of the monitoring system has been attacked, or if the credibility of the monitored information is weak? These questions lead us to believe that derivation of monitoring from SLAs is not a simple straightforward process. Rather, **monitoring has to be adaptive in order to satisfy high level objectives** coming from the functional QoS, monitoring QoS, and even other FCAPS objectives. To tackle these issues, we defined a 3-layered adaptive monitoring framework [3]: the *configurability* layer defines monitoring operations (such as *polling* or *listening*), the *adaptability* layer specifies operators applying on the previous layer, while the *governability* layer represents the "intelligence" of the adaptation by invoking the operators of the adaptability layer. Since the first two layers have been respectively described in [16] and [14], we focus in this paper on the governability layer and propose a methodology to specify the policies required to guarantee both functional and monitoring QoS requirements. The paper is organized as follows: the next section gives an overview of some existing approaches and points out their weaknesses; section 3 briefly exposes our adaptive monitoring framework so that our methodology can be introduced in section 4; before concluding, a use case illustrating the monitoring adaptation of a cloud provider is presented.

II. Related Work

The studied research works lead us to raise three contribution trends: monitoring SLA, autonomic management and adaptive monitoring.

Monitoring SLAs is not a trivial task, especially when they don't match with predefined templates. [4] highlights the complexity of SLA transformation -using Finite State Machines- to introduce an executable unambiguous contract that could be monitored during the SLA life-time, while [5] discusses four approaches and two concrete architectures that could be adopted to collect SLA metrics.

Regarding autonomic management, [6] suggests a mapping between "*high level*" metrics (found in SLA) and "*low level*" metrics (fetched by soliciting agents) to "*facilitate autonomic SLA management and enforcement*". [2] proposes a framework where automatic monitoring (add/remove services metrics) is derived from contracts and related to cross-organizational workflow. [7] proposes an autonomic frame-

work that monitors QoS satisfaction on both the service and infrastructure levels; identifying the relevant metrics, as well as the right time intervals, can be performed on-the-fly, while adaptations are guided by "*end-user preferences and predefined application or platform policies*".

Regarding adaptive monitoring, [8] proposes an automated monitoring Web Service that could be deployed and reconfigured at run-time. The monitoring reconfiguration relates on operations such as adding new monitoring instances/deleting an existing one, or scaling/withdrawing of monitored resources. [9] introduces an architecture composed of two levels of managers (*main* managers are responsible for planning and deploying monitoring, while *intermediate* managers solicit SNMP agents) where monitoring adaptability is achieved by the replacement of failed managers. [10] adapts monitoring in order to keep it minimal, but extends it when a SLA metric violation occurs. [19] is concerned by the detection of SLA violations in large networks, where adaptation actions are taken in order to determine subsets of end-to-end paths.

Thus, most works related to QoS monitoring focus either on (1) the auto-configuration of SLA-oriented monitoring, or (2) the reconfiguration of the functional system based on the monitored metrics, or (3) the reconfiguration of the monitoring itself, starting from either explicit demands or static objectives. But none of these approaches offers an accurate visibility on the quality of the metrics values and/or their instrumentation mechanisms. In fact, managements decisions are directly influenced by the quality of monitoring, thus we argue that increasing the QoS of the whole system requires (begins from) designing adaptive monitoring system that ensures quality constraints, instead of monitoring quality metrics only.

In this paper, our proposal tries to tackle these major issues by making monitoring self-managed regarding high level quality goals, while taking into account the functional QoS & monitoring quality requirements.

III. The Adaptive Monitoring Framework

Our approach is based on a 3-layered framework illustrated on Figure 1, and defines three fundamental capabilities required to control monitoring: being **configurable**, **adaptive** and **governable**. [3][14][16] introduce an adaptive monitoring service where monitoring activities become independent of any consideration regarding agents or management protocols: the originality of this approach relies on its capacity of hiding the complex integration of heterogeneous target agents (entities providing management data) and management protocols (used to communicate with those agents).

The **configurability layer** stands on the DMTF Common Information Model (CIM) standard. This low level layer aims at representing, in addition to the systems to be managed, the metrics and monitoring mechanisms that are required to ensure both functional and monitoring QoS; this layer operates these mechanisms as well.

The classes *IRIT_PollingOperation and IRIT_ListeningOperation* respectively specify the polling and event

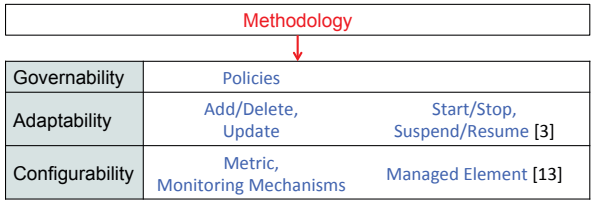

Fig. 1: An Adaptive Monitoring Framework

reporting mechanisms. The polling operation can be configured according to various properties (*PollPeriod*, *RequestDelay*, *MaxIteration*, *TemporalValue*) and statistics (*UnsuccessfulOperationRate*, *UnsuccessfulOperationThreshold*) that guide the autonomous behavior of polling. As this paper does not deal with event reporting, we do not mention their configurability. In practice, the framework performs polling by soliciting "targets" defined as managed elements or sources of pulled data (i.e. remote agents).

In addition, this monitoring model is associated with a metric model to derive the configuration of the monitoring mechanisms based on the QoS specification (i.e. QoS metrics & constraints to be monitored). This derivation is feasible by virtue of our CIM Model extension [13] that classifies metrics according to their instrumentation: (i) *Resources Metrics* are instrumented with pulled data, thus they are associated with *IRIT_PollingOperation* instances, (ii) *Measurable Metrics* are processed programmatically using algorithms (they are not gathered distantly), and (iii) *Mathematical Metrics* are calculated based on formulas applied to elementary metrics.

The **adaptability layer** provides "operators" that apply on the lower layer. Using these operators, some instances of monitoring services can be added, deleted or updated (by modifying their configuration), and the underlying mechanisms (polling and event reporting) can be started, stopped, suspended or resumed.

Finally, the **governability layer** is the top level layer representing the "intelligence" of the monitoring adaptation. To express both functional and monitoring objectives, it uses policies to describe *when* and *how* adaptation should take place, that is when and where operators of the adaptability layer should be invoked. Considering this adaptable monitoring framework, a question arises: how to build the governance level in order to monitor SLAs from both the functional and monitoring points of view? This issue could be driven by a top-down approach that help to specify "high level objectives", and then derive these objectives through the configuration and adaptation of monitoring operations.

IV. A Goal-Oriented Approach for Adaptive SLA Monitoring

Thinking that one of the existing software engineering approaches should answer our needs, we looked forward a suitable method for designing moniroting adaptation. The origin of Requirements Engineering (RE) goes back to the need to avoid crucial mistakes committed at the project design phase,

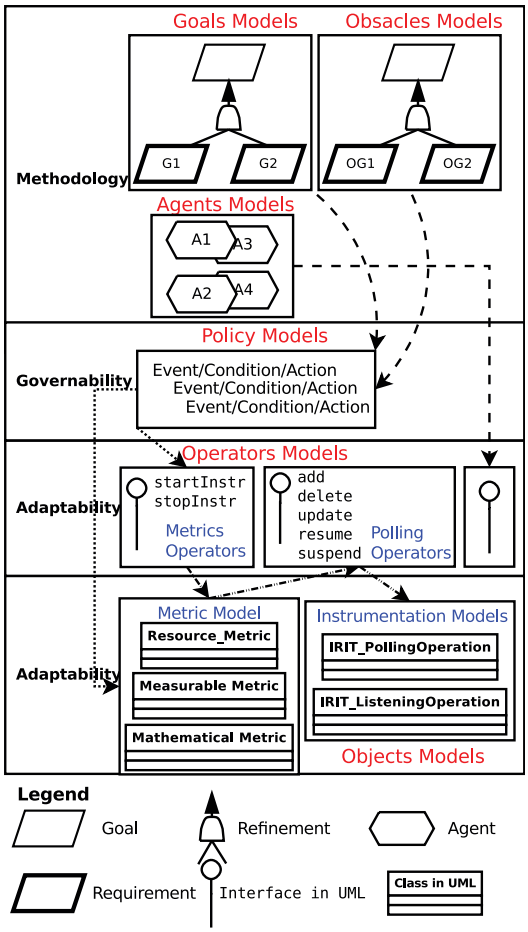

Fig. 2: Our Goal-Oriented Approach for Adaptive SLA Monitoring

and aims at building systems solving real-world problems. This methodology applies iterative activities about "*eliciting, evaluating, documenting, consolidating and changing the objectives, functionalities, assumptions qualities and constraints that the system-to-be*[1] *should meet based on the opportunities and capabilities provided by new technologies*" [11]. Among multiple RE approaches, the KAOS[2] method is goal-oriented and provides a formal assertion layer that proves correctness and completeness of goals [12].

A. Our Goal-Oriented Approach for Adaptive SLA Monitoring

The resulting methodology is illustrated on Figure 2, and follows a top-down approach to fit the management area specificities. We focus here on the **self-management of monitoring** integrated into autonomic systems, but our methodology applies to any management system handling any FCAPS feature.

The **goal models** elicit "goals" representing high level behavioral prescription of the system-to-be (the functional or monitoring systems in this study). Each goal expresses one or more objectives, and a given goal is completed through the cooperation of several components or actors, the so-called agents[3]; as goals belong to two distinct systems, we distinguished two categories of agents: *Functional System QoS Agents*, and *Monitoring System QoS Agents*. Goals are decomposed into sub-goals via AND/OR refinement methods, where the most refined goals are called *Requirements*, or *Leaf Goals*; in KAOS, each of those leaf goals are assigned to a specific agent in order to be realized. Within our framework, they match with objectives related to the QoS management of both the functional system and the monitoring system itself.

Others goals, depicted within the **obstacle models**, are deduced from the goal models and may prevent the satisfaction of these lasts. Since the goal models rely on metrics gathered by the monitoring system, the obstacle models comprise goals related to the quality of those metrics. Such goals include, among others, degradations of monitoring: if those degradations occur (if obstacle goals are satisfied), they cause the non-satisfaction of some QoS management goals.

Starting from both the goal and obstacle models, the **policy models** specify events triggering the evaluation of conditions and leading, when conditions are verified, to the execution of some specific adaptation actions. The policy models thus ensure that goals of the functional and monitoring systems remain satisfied, whereas obstacle goals keep unsatisfied. Let us note that in fact, adaptation actions match with leaf goals (or requirements).

Finally, from the policy models, the **operation** and **object models** are designed. Indeed, the event and condition specified within a given policy relies on one or several metrics or managed elements defined into the configurability layer; the object models thus define objects representing the system-to-be resources, their dependencies and their topology, together with the metric model [13] and monitoring mechanisms [3] presented in the previous section. On the other hand, the **operation models** are composed of the sets of operations required by agents to achieve the *Leaf Goals*, that is the actions specified into the policy models. They match with the operators defined within the adaptability layer, and acts on the monitoring services specified in the object models.

This methodology points out the fact that the identification of leaf goals represents a crucial step in this approach: from the requirements, adaptation policies can be specified and, from here, the models of the configurability and adaptability layers are built. Therefore, to facilitate this identification process, we suggest a classification of leaf goals according to the *dimension* they belong to.

B. A Classification of Leaf Goals

We conducted a reflexion about the "factors" a monitoring adaptation action may apply to. From this study, we identified four categories of leaf goals, or *dimensions*, illustrated on

[1]The "system-to-be" refers to the system with RE machine operating in it.

[2]KAOS stands for "Knowledge Acquisition in Automated Specification", or "Keep All Objectives Satisfied".

[3]Notice that *Agents* in networks and systems Management are entities responding to Management requests coming from other Management entities called *Managers*; therefore the term "*Agent*" in RE has a different meaning.

Figure 3: Spatial, Metric, Temporal, Gathering. For clarity reasons, we illustrated each dimension with examples of leaf goals that may be reached within our framework, but this list is far from being exhaustive: the aim of the dimensions is to make the design of policies (and thus, the design of operation and object models) easier and more efficient for human administrators.

The **spatial dimension** comprises goals focusing on the target number of managed elements composing the functional or monitoring systems. When new entities (dis)appear within those systems, (less)more targets have to be monitored, and the monitoring system has to self-adapt accordingly: it has to (expand)shrink its monitoring perimeter. We thus defined two leaf goals into this dimension: *Expand Monitoring Perimeter*, and *Shrink Monitoring Perimeter*.

The **metric dimension** implies goals focusing on the number of monitored metrics. This amount of metrics may evolve according to a negociation of the SLA (i.e. additional metrics have to be monitored to meet the new SLA requirements), or to a modification of the monitoring perimeter (the number of monitored metrics increases/dicreases when the monitoring perimeter expands/shrinks). Therefore, the leaf goal *Zoom In* has been identified to monitor more metrics, whereas the leaf goal *Zoom Out* allows to stop monitoring a given (set of) metric(s).

The **temporal dimension** relates on goals focusing on temporal behavior. We distinguish two levels of temporal granularity: the fine-grained level deals with individual temporal characteristics, whereas the coarse-grained level addresses collective temporal behavior. Within our framework, this dimension includes three leaf goals related to the polling mechanism and illustrated on Figure 3: the *Update Period* to update the frequency of a given polling, the *Align Polling* to launch concurrent pollings at the same time, and the *Disalign Polling* to process pollings according to a given offset.

The **gathering dimension** deals with goals focusing on the manner of getting/exchanging information. Such goals occur when the way of getting a desired metric has to be updated (i.e. another target has to be invoked to get a given metric value, or another communication protocol has to be used to invoke a given target). The matching leaf goals we identified are *Alter Target Entity* and *Alter Target Protocol*.

Our analysis showed up that the adaptation of monitoring implies some sort of ”semantic”. We believe that handling adaptation with more coarse granularity expressing its semantic, by adopting a goal-oriented approach, makes monitoring adaptation more suitable to fit quality of measures' requirements. Moreover, goals can be classified into distinct *dimensions*, thus providing a new ”starting point” reflection to establish a panel with a variety of adaptation actions.

V. Case Study: a Cloud Provider

In order to show the applicability of the proposed approach, we address here an example related to the adaptation of the monitoring system (not the functional system).

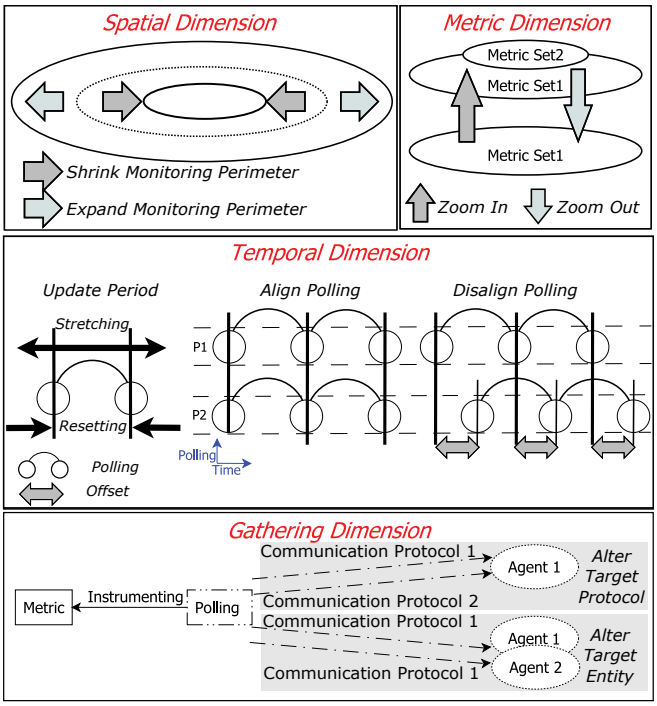

Fig. 3: Dimensions and Leaf Goals

Our scenario rolls in a cloud data center hosting a large number of virtual machines (VMs) created/deleted on demand, and which provides its clients with a service offering continuous monitoring of some agreed SLAs metrics. When the number of VMs is too high, the monitoring service might not be able to respect the determined metrics freshness, and thus breaks the SLA agreement.

To tackle this issue, the cloud provider has to adapt its monitoring. Two high level goals to be satisfied during the monitoring system runtime have been identified: *Respect Metrics Freshness* makes sure that the SLA is respected, and *Minimize Monitoring Cost* aims at dicreasing the consumption of resources dedicated to monitoring as much as possible. The former goal can be decomposed by OR-refinement into two leaf goals (see Figure 4): *Shrink Monitoring Perimeter* to decrease the load of the monitoring service, and *Alter Target Entity* to retrieve the metric, in time, from another target. On the other hand, the goal *Minimize Monitoring Cost* matches with the leaf goal *Expand Monitoring Perimeter*, to monitor ressources handled by others monitoring services so that they are able to stop their activity (or even shudown).

The matching adaptive monitoring framework (based on the two above high level goals) has been first simulated and then implemented. However, we focus here on the *Shrink Monitoring Perimeter* requirement.

A. Shrink Monitoring Perimeter

We adopted a distributed management architecture to process the *Shrink Monitoring Perimeter* leaf goal, and introduced two types of managers as CIM servers hosting the moni-

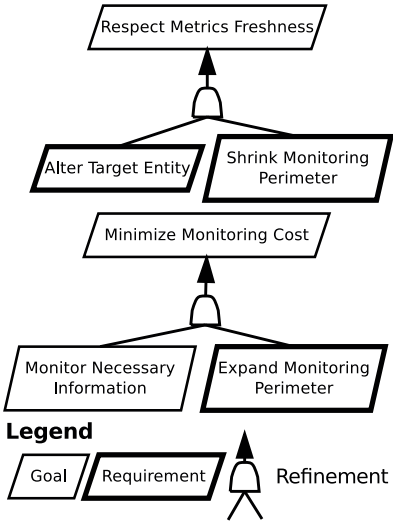

Fig. 4: Use Case Goal Refinement

toring service defined in [3]: the *Primary Managers* (PM) are responsible for life-cycle management of SLAs, whereas the *Assistant Managers* (AM) act as temporary managers to dicrease the load on a given PM (each primary manager has its own pool of assistants). Then, the adaptation action consists in transferring some resources managed by a primary manager to one or more of its assistants, so that the interval period between two polls for the same metric becomes lower than a given treshold.

This operation is handled by an additional component, called agent (see Section IV-A), responsible for instrumenting the leaf goal. The agent executes the following sequence of process once it receives the stimuli from the PM:

1) It randomly picks up and launches an inactive assistant manager.
2) It randomly picks up a set of target instances to handle; the set size is specified through a dedicated parameter (see further).
3) It forwards the selected set of instances to the launched assistant manager. Starting from this moment, pollings to the matching targets are operated by the AM.
4) Finally, once it receives an acknowledgement from the AM, it deletes the delegated instances within the primary manager.

B. Simulation

The above scenario has been simulated using Colored Petri Nets (CPN) [17][20] and CPN Tools V3.5.7. CPN is a modeling language for concurrent and reactive systems. The latter leverage on communication, synchronous and asynchronous events to collaborate. It produces formal models, due to the mathematical definition of its syntax and semantic. Those models are "executable", that is, they could be simulated, in order to illustrate visually the evolution during the system runtime. Simulation is useful to detect early misconception at the design phase. Because of the formal nature of CPN models, state space process calculates all reachable states of the model, therefore those models could be the subject of some "verification and validation questions" to ensure some desired properties regarding the system-to-be. Beside graphical capabilities of creating models, CPN involves a functional programming language, namely CPN ML, which is based on the Standard ML, providing primitives for defining data types and writing treatment functions.

The simulation has been initialized according to the following parameters: *metric freshness* was set to 20 time unit; the *inter-arrival rate* of *Poisson* distribution regulating the arrival of VMs in the cloud is $\lambda = 1/60$, (i.e. the time between two arrivals has a mean of 60 time unit); the *service rate* of *Erlang* distribution representing the *polling service time* is $\mu = 1$ (i.e. the time to serve polling has a mean of 1 time unit). In addition, the *number* of targets delegated by the server to a given AM is 50, and this process occurs when the *polling service time* exceeds the metric freshness (20 time unit) more than fifty times ($\mathbb{N}$=50).

Figure 5 illustrates measurements of the *polling service time* belonging to two simulation replications. In these simulations, VMs are continuously created and joined to the monitoring perimeter of PM. In addition, simulations are automatically stopped after generating 250 VMs. During simulation, the *Metric Evaluator* continuously evaluates the *polling service time* of each polling operation (instrumenting the SLA template metrics), against the determined threshold. The latter, in our use case is set to be equal to the metric freshness. Once the *polling service time* is violated more than fifty times, the *Shrinking* process will be fired up as much as necessary. In each of these two simulations, PM performs *Shrinking* three times (vertical bars in Figure 5 represent the moments of time where *Shrinking* is performed). That is due to the continuous creation of VMs, as well as the violation of *polling service time*; that is decreased significantly after each *Shrinking*. At the end of the simulation, we can notice that AM1, AM2, and AM3 are launched. Each of them monitors 50 targets according to their appropriate 100 pooling operations. Furthermore, 100 targets and their 200 associated polling operations are still monitored by the PM. We purposely continue the simulation after the launching of all AMs, to illustrate the increasing of the *polling service time* regarding PM remaining polling.

By simulating our use case, we have been capable of illustrating (using a formal-language based tool) the importance of monitoring adaptation, when this monitoring is performed regarding quality concerns (Qos & QoI). As well as, by analyzing simulation measurements, we were able to show that semantic behind goals (which guided us in goals refinement process), can concretely take place by goal realization.

VI. Conclusion & Perspectives

We have proposed a goal-oriented approach for designing adaptive SLA monitoring. This approach guides the management expert to express the monitoring behaviour starting from any SLA contract. It enhances to focus on SLA-monitoring

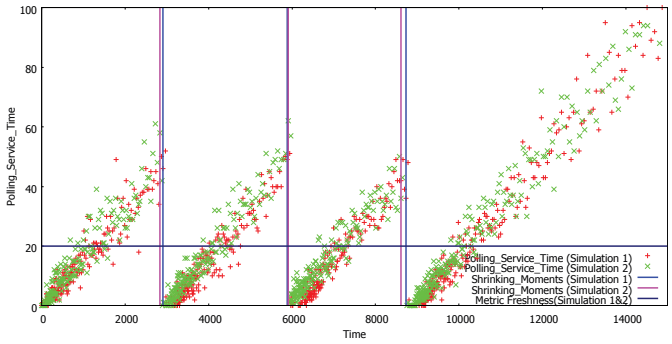

Fig. 5: Simulation Measurements

adaptation in regard with: 1) the managed system changes and its related functional requirements, 2) the non-functional requirements, and 3) violation detection.

It links the defined monitoring adaptation (dimensions) to several models that drive the distributed monitoring execution. As examples, we have determinated (1) the adaptation triggers (by instantiating metrics & constraints instances, and then instrumenting and evaluating them), (2) the adaptation policy (by instantiating adaptation policy, and automating the subscription to constraints violations).

Due to the 3-layers adaptable monitoring framework, this approach hides complexity and heterogeneity of candidate management protocols by focusing on monitoring adaptation behaviour. This approach benefits from formal CPN notation and simulation for validating the designed adaptative monitoring behaviour. An implementation of this simulation on a specific-technology platform has just been achieved, but we didn't conduct any experimentation yet; however, some results should be discussed by the time of the cofnerence.

The main perspectives we currently investigate are the followings: 1) the specification of complex goals (composed of leaf goals) 2) support for the designing process 3) the adaptive monitoring as a service

REFERENCES

[1] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing," *Computer*, vol. 36, no. 1, pp. 41–50, Jan. 2003.

[2] P. Grefen, K. Aberer, H. Ludwig, and Y. Hoffner, "Crossflow: CrossFlow: Cross-Organizational Workflow Management in Dynamic Virtual Enterprises," *International Journal of Computer Systems Science & Engineering*, vol. 15, pp. 277–290, 2000.

[3] A. Moui, T. Desprats, E. Lavinal, and M. Sibilla, "A CIM-based Framework to Manage Monitoring Adaptability," in *Network and service management (cnsm), 2012 8th international conference and 2012 workshop on systems virtualiztion management (svm)*, 2012, pp. 261–265.

[4] C. Molina-jimenez, C. Molina-jimenez, S. Shrivastava, S. Shrivastava, E. Solaiman, E. Solaiman, J. Warne, and J. Warne, "Run-time Monitoring and Enforcement of Electronic Contracts," *Electronic Commerce Research and Applications*, vol. 3, p. 2004, 2004.

[5] C. Molina-Jimenez, S. Shrivastava, J. Crowcroft, and P. Gevros, "On the Monitoring of Contractual Service Level Agreements," in *Electronic Contracting, 2004. Proceedings.*, 2004, pp. 1–8.

[6] V. Emeakaroha, I. Brandic, M. Maurer, and S. Dustdar, "Low Level Metrics to High Level SLAs - LoM2HiS Framework: Bridging the Gap Between Monitored Metrics and SLA Parameters in Cloud Environments," in *High Performance Computing and Simulation (HPCS), 2010 International Conference on*, 2010, pp. 48–54.

[7] G. Katsaros, G. Kousiouris, S. V. Gogouvitis, D. Kyriazis, A. Menychtas, and T. Varvarigou, "A Self-adaptive hierarchical monitoring mechanism for Clouds," *Journal of Systems and Software*, vol. 85, no. 5, pp. 1029 – 1041, 2012.

[8] D. Roxburgh, D. Spaven, and C. Gallen, "Monitoring as an SLA-oriented consumable service for SaaS Assurance: A Prototype," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, 2011, pp. 925–939.

[9] P. Thongtra and F. Aagesen, "An Adaptable Capability Monitoring System," in *Networking and Services (ICNS)*, 2010, pp. 73–80.

[10] M. Munawar, T. Reidemeister, M. Jiang, A. George, and P. Ward, "Adaptive Monitoring with Dynamic Differential Tracing-Based Diagnosis," in *Managing Large-Scale Service Deployment*, ser. Lecture Notes in Computer Science, F. Turck, W. Kellerer, and G. Kormentzas, Eds. Springer Berlin Heidelberg, 2008, vol. 5273, pp. 162–175.

[11] A. van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley, 2009.

[12] A. van Lamsweerde, "Requirements Engineering in the Year 00: A Research Perspective," in *Proceedings of the 22nd international conference on Software engineering*, 2000, pp. 5–19.

[13] A. Toueir, J. Broisin, and M. Sibilla, "Toward Configurable Performance Monitoring Introduction to Mathematical Support for Metric Representation and Instrumentation of the CIM Metric Model," in *Systems and Virtualization Management (SVM), 2011 5th International DMTF Academic Alliance Workshop on*, 2011, pp. 1–6.

[14] A. Moui, T. Desprats, E. Lavinal, and M. Sibilla, "Information Models for Managing Monitoring Adaptation Enforcement," in *International Conference on Adaptive and Self-adaptive Systems and Applications (ADAPTIVE), Nice, 22/07/2012-27/07/2012*, D. Petcu and E. Troubitsyna, Eds. Xpert Publishing Service (XPS), juillet 2012, pp. 44–50.

[15] C. Hobbs, *A Practical Approach to WBEM/CIM Management*. Taylor & Francis, 2004.

[16] A. Moui, T. Desprats, E. Lavinal, and M. Sibilla, "Managing polling adaptability in a CIM/WBEM infrastructure," in *Systems and Virtualization Management (SVM), 2010 4th International DMTF Academic Alliance Workshop on*, 2010, pp. 1–6.

[17] K. Jensen and M. Lars Kristensen, *Coloured Petri Nets Modelling and Validation of Concurrent Systems*. Springer, 2009.

[18] J. Jørgensen, S. Tjell, and J. Fernandes, "Formal requirements modelling with executable use cases and coloured Petri nets," *Innovations in Systems and Software Engineering*, vol. 5, no. 1, pp. 13–25, 2009.

[19] J. Nobre, L. Granville, A. Clemm, and A. Prieto, "Decentralized Detection of SLA Violations Using P2P Technology," in *Network and service management (cnsm), 2012 8th international conference and 2012 workshop on systems virtualiztion management (svm)*, 2012, pp. 100–107.

[20] K. Jensen, L. M. Kristensen, and L. Wells, "Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems," *Int. J. Softw. Tools Technol. Transf.*, vol. 9, no. 3, pp. 213–254, May 2007.