Changeloads: a Fundamental Piece on the SASO Systems Benchmarking Puzzle

Position Paper

Raquel Almeida and Marco Vieira
CISUC, Department of Informatics Engineering
University of Coimbra
Coimbra, Portugal
{rrute, mvieira}@dei.uc.pt

***Abstract*—Benchmarks have been traditionally tailored to static, unchangeable systems, functioning in well-known and controlled environments. Thus, established benchmarks (and benchmarking approaches) are becoming progressively less representative of real world scenarios, as change is gaining emphasis as a fundamental player in computing systems runtime conditions. As today's systems are becoming, at the very least, reactive to changes (either endogenous or exogenous) at some level, if not even proactive in reaching their goals more efficiently and effectively, we believe that benchmarks must also evolve, becoming applicable to systems that react to change, adapt, evolve, and have the capability to improve their own performance. In this position paper, we argue that representative changeloads are now as vital as representative workloads, and that changeload-based benchmarks will become a key part in the development and evaluation of SASO systems. We present and discuss some applications of benchmarks in this area, proposing some directions for research.**

Keywords- benchmarking; changeload; resilience; performance; self-adaptive; self-organizing; autonomic systems

I. INTRODUCTION

Over the last decades, the computing systems world has changed: computers are now expected to proactively deal with faults and failure in components, exhaustion of resources, user errors, and are immersed in changing environments, facing changing user needs, peak load problems, or insufficient bandwidth in networks, just to name a few [1]. In the near future, autonomic characteristics will help systems dealing with their own increased complexity, and that of the surrounding environment. Unfortunately, the provision of flexibility and the ability to handle changes during runtime comes at the price of reducing the predictability the impact of adaptation on the system itself, especially with respect to its non-functional requirements. How can one assess if a system will maintain expected dependability or availability at runtime in spite of changes that may occur in the system or in its environment? Is there a way to determine if a particular alternative for adaptation is better than another, or that a particular repair will not make the situation worse? Although major advances have been made, existing approaches in self-adaptation do not systematically address the need to determine if an adaptive system can deliver a service that can justifiably be trusted when facing changes (i.e., that it will be resilient [2]). This lack of assurances is a critical issue, as it may end up hindering the widespread adoption of self-adaptive and self-organizing (SASO) systems, often regarded as not being dependable and reliable by the final users [3].

Established benchmarking approaches seem to fall short of what is expected from them when evaluating and comparing adaptive systems. In fact, over the last decades benchmarks have been tailored to unchangeable and manageable systems, predominantly running in static (slow-changing) or predictable environments. The problem is that this approach is becoming ever less representative of real world scenarios, as change further becomes a fundamental player in runtime conditions for computing systems. In practice, well-known performance benchmarks and metrics are no longer sufficient for describing all relevant aspects related to today's systems, as qualities like reliability, availability, energy consumption, scalability, among others, have gained more relevance in many contexts. Even if we maintain the focus on performance evaluation, we need new processes to describe and assess the ability of systems to solve runtime-specific problems, as they deal with variability and unexpected events. Change is now a constant, if not of the system itself, at least of its executing environment in the real world, and this will surely impact its runtime performance. Furthermore, vendors and users will soon need ways to compare previous state-of-the-art systems to new systems with self-* characteristics in a fair head-to-head evaluation, so that good and reliable insight into the cost-to-benefit relationship of using the new proposed solutions is provided.

In this position paper, we argue that a new construct is required in the evaluation and comparison of adaptive systems: a *changeload*. As a significant and adequate workload (typically a representative use-case within a domain) is fundamental for a reliable and sustained evaluation and comparison of systems, by allowing the examination of the system response to the type and amount of work that will be typically expected of it, so must be a representative changeload. Such an element will submit the system under evaluation to relevant stimulus (potential changes that might trigger adaptation mechanisms of particular interest) to probe and assess how the system responds to them. In a way, while a workload simulates the service stress that a system may face in the field, the changeload would be responsible for modeling the variations and fluctuations in the overall stress (either from an endogenous or exogenous source), providing a more realistic execution use-case. As adaptation mechanisms are aimed at

This work is partially supported by the ADAAS - Assuring Dependability in Architecture-based Adaptive Systems project, in the context of the CMU-Portugal program.

endowing systems with abilities to deal with fluctuations and changes, we argue that representative changeloads are the best way to accurately evaluate their relevance and effectiveness, and their impact on the service provided as perceived by the user.

The outline of the paper is as follows. The next section discusses how adaptive computing systems are emphasizing the importance of characterizing the systems' prospective executing scenarios. Section 3 discusses issues and current research challenges in the definition of changeloads for the evaluation of adaptive systems. Section 4 presents some directions for benchmarking systems with self-adaptive characteristics and features, proposing two examples of changeload-based benchmarking use in the evaluation of SASO systems. Finally, Section 5 concludes the paper, presenting our view regarding the future of benchmarking and the importance of more investment in changeloads research.

II. The Prominence of Prospective Execution Scenarios in SASO Systems development & evaluation

A. A decisive player in runtime execution scenarios: Change

Highly complex infrastructures, composed of heterogeneous applications running on a variety of platforms and operating systems, combining hundreds of system components, with thousands of tuning parameters, are now a reality. Furthermore, today's information society creates highly variable and somewhat unpredictable workloads on those interconnected, distributed, heterogeneous systems, and more than ever many business models depend on their IT infrastructure being available and performing its best 24/7. System components and software have to evolve to deal with the increased complexity of control and operational management [4], and existing paradigms based on static requirements, interactions, compositions and behaviors are becoming no longer suitable [5], as industry and researchers acknowledge the need for development and optimization (at design time and runtime) for changing runtime settings. As a consequence, the importance of evaluation of systems characteristics and quality attributes under changing conditions is gaining strength.

We believe there is consensus in recognizing that the most determining factor for success of SASO systems is their capability to operate and adapt under dynamic circumstances, which allows them to provide the expected service and attain their goals (captured as functional and non-functional requirements) as best as possible, during their lifetime. As a result, the prospective execution scenarios of an adaptive computing system gain extreme relevance, as they are determinant in guiding decisions during both development and runtime, in an effort to provide the system with the adequate qualities and self-* characteristics that will empower it to effectively and efficiently face changing runtime circumstances and settings. In light of this, we argue that better knowledge and modeling of prospective execution scenarios is imperative.

We propose the characterization of an *execution scenario* in terms of state (system and environment), system goals, and changes potentially applied to that state. An execution scenario is then a tuple (wl, oc, G, C), where: *wl* represents the workload (amount and type of work assigned to the system); *oc* are the operational conditions of the system (including software and hardware resources needed for the system to perform its service); *G* is a set of system goals; and *C* is a set of changes that the state determined by the workload and operational conditions may face.

In the definition above, goals represent the quality attributes that the system should fulfill at runtime (e.g., maximum response time, minimum throughput, minimum availability, etc.), including a prioritization among potentially conflicting goals. Much research work has been dedicated to characterizing and modeling workloads and their fluctuations [6–8], but the major source of variations in execution settings is the set of potential changes in operational conditions (e.g., fluctuations in network performance, software updates or faults, variations in available resources, services or devices) and system goals (e.g., requirements modifications, different user preferences). We believe that there are many research paths yet unexplored in this direction.

B. Change Scenarios and Changeloads

Considering the execution scenarios as defined above, we may further distinguish between conventional scenarios, in which the system is executing without experiencing any anomalies, and scenarios associated with changes in the system or its environment that induce anomalies in the system, typically triggering adaptations. These can be designated as *base* and *change* scenarios, respectively. Here, we present a refined definition of these concepts, firstly introduced in [9].

A *base scenario* corresponds to the most common conditions of execution of the system. The workload should be representative of the typical amount and type of work assigned to (or expected from) the system in a specified time period, and operation conditions would comprise a typical setup of systems in the domain (including hardware and software resources typically used), as well as a representative characterization of the system's environment. Hence, a base scenario reflects the operational characteristics of systems in the domain while running a typical workload and operating in the absence of changes, setting the baseline for comparison with situations when the system is faced with changes that may drive it into an adaptation process. It should be noted that by "typical" we do not mean static: the base scenario represents a *stable* state of the system with no abnormal or faulty conditions considered.

Change scenarios are based on a base scenario, but include a representative sequence of changes that may affect the system and its ability to achieve and maintain the fixed goals specified in the base scenario. It can then be defined as a tuple (wl_t, oc_t, G, C), where: wl_t represents the typical workload; oc_t represents the typical operational conditions of the system; *G* is a set of system goals; and $C \neq \varnothing$ is a set of changes applied to the state determined by the workload and operational conditions. A change scenario is then defined by a typical condition of the system followed by a non-empty set of changes, while a *changeload* is a set of change scenarios.

In this perspective, changeloads become decisive in the evaluation of adaptive systems, both during design and

execution phases. They can provide a realistic mean to assess if adaptation is being achieved in an efficient and effective manner by subjecting the system to representative changes and evaluating the systems response, enabling better well-informed and substantiated choices among the existing adaptation strategies, algorithms and architectures. Moreover, representative changeloads subjected to refinement based on system execution and adaptation history at runtime may be instrumental in allowing a more sustained assessment (and eventual review) of adaptation strategies (either devised at design or runtime), as well as adaptation decisions taken by the system itself during execution.

III. CHARACTERIZING AND DESIGNING CHANGELOADS

The first step towards the definition of a representative changeload that can be used to evaluate and compare alternative adaptation techniques and strategies or alternative adaptive systems is determining which changes are more bound to affect the system ability to attain its goals (eventually triggering adaptations). The systematic identification and classification of change types is then fundamental to support the definition of change scenarios, as these rely on the instantiation of types of changes for the considered application domain, system type, and prospective execution settings.

A. Identifying and characterizing changes

The number of potential changes that the system and its environment can go through at run-time is virtually unbounded. Hence, we need to focus on identifying only the relevant stimulus, the ones that may trigger adaptation mechanisms of particular interest, and thus have the best potential of relevance for the evaluation of how the system responds and behaves under varying conditions and settings.

We define a *change type* as a tuple (src, A, B) that characterizes a change, where: *src* identifies the source of the change: environment (including human operators), resources used by the system, or the system itself; $A = \langle a_1,...,a_k \rangle$ is a vector of attributes that hold information about the specific properties (variables) associated with the change type; and $B = \langle b_1,...,b_k \rangle$ describes the dynamics of the attributes in A (how they evolve over time, e.g., through a polynomial, exponential, or step function).

Research effort should be directed towards the identification, collection and systematization of types of changes that might impact and induce significant stress to adaptive systems in distinct operational and application domains, aiming at the compilation of change taxonomies for different areas of SASO systems. These could then be used as a starting point for the selection and instantiation of changes to be included in change scenarios and changeloads for specific systems and application domains.

We believe that one promising approach to accomplish the identification of representative change types is to use historical data and logs to correlate basic system indicators and resources fluctuations (e.g. CPU, memory and disk usage, network traffic, system response time, etc.) with identifiable service level events, resulting on the identification of change instances faced by systems in real situations. Our assumption is that even though some system mechanisms may mask service level impact of some changes, this impact is still identifiable in the basic resources indicators. We are presently working on evaluating the validity of this approach by analyzing one year worth of indicators collected from a server DBMS within an academic context, in an effort to identify change instances that impacted the system, and eventually classify them into representative change types for this type of systems based on data from the real world.

B. Defining and selecting representative changeloads

Each possible change scenario for the systems in the evaluation domain is characterized by a base scenario and a set of representative changes that may affect the system's ability to achieve and maintain its goals (i.e., as specified in the base scenario). Each change included in the change scenario is an *instantiation* of a general change type (as defined in III.A). Given a set of change types CT, a change is a tuple (ct, src_{inst}, A_{inst}, B_{inst}, ti, d) that corresponds to an instantiation of a change type, where: ct = (src, A, B) ∈ CT determines the change type to be instanced as a change; src_{inst} is the instance of the source of change (i.e., where it actually occurs); $A_{inst} = \langle a_{1inst},...,a_{kinst} \rangle$ is a vector of attribute values instantiating the attributes in A; $B_{inst} = \langle b_{1inst},...,b_{kinst} \rangle$ of behavior instances of the elements in B; *ti* determines the time instant in which the change instance is triggered; and *d* is the duration associated with the change.

The trigger instant assigned to each specific change instantiation determines the sequence of changes in a change scenario. Also, the combination of trigger instant and duration for different changes may result in a combined effect of changes on the system. It is worth observing that while some specific changes may be transient, impacting the system during a particular amount of time (e.g., a temporary peak in workload), the duration can be considered equal to ∞ if the change is permanent, or 0 if non-applicable (i.e., instantaneous change).

Structuring and identifying changeloads of interest for a particular area of application or type of SA or SO system presents several significant challenges, which include, but are not limited to:

1) Selecting specific changes: for each underlying context, the relevant changes must be identified. This challenge is related to the second.

2) Reducing the change space: systematized approaches must be devised to deal with the exponential growth of the number of possible changes that might afect the system.

3) Identifying relevant sequences of changes: combining distinct change events in a same change scenario seems imperative for real-world situations representativeness, but also presents itself as a complex and difficult task.

4) Scheduling and timing: deciding on triggering instant and duration for each change, as well as orchestrating the execution of changeload with the workload during system evaluation also poses significant challenges.

In [9] we have proposed an approach based on risk analysis to identify and select the most realistic and relevant (sequences of) changes (i.e., change scenarios) to be included in resilience

benchmarking of self-adaptive systems, but this approach mostly tries to tackle challenges 1) and 2).

IV. BENCHMARKS FOR SASO SYSTEMS

In a previous work [10], we presented and discussed some research challenges and perspectives on the design of benchmarks for self-adaptive systems. Although mainly intended for the evaluation of resilience and comparison of similar self-adaptive software systems (i.e., systems implementing similar functional requirements and intended to execute in comparable environments), the ideas and concepts presented, specifically referent to benchmark components, can be extended and refined to be applicable in diverse evaluation situations for SASO systems. Two examples of application of changeload-based benchmarks are given in the following subsections, illustrating the potential relevance and applicability of such benchmarks in distinct phases of the lifecycle of an adaptive system.

A. Comparing adaptive solutions for the same problem

As research on SASO systems advances, developers need validated and reliable ways to evaluate and compare alternative approaches during the system development lifecycle. The availability of changeload-based benchmarks, incorporating procedures and metrics to reflect timeliness of adaptation, overheads, robustness of strategies, among others, would be a key tool to realistically evaluate different versions of a system. We are working on an experimental approach to evaluate and compare diverse adaptation strategies for a self-adaptive software system, using as case study Rainbow (architecture-based platform for self-adaptation) and Znn.com. This work will rely on the identification of representative changeloads, which will be used in a run-time stimulation of the system.

B. Comparing adaptive systems aimed at providing the same service

Evaluation and comparison of SASO systems resiliency, which we believe is the most relevant aspect from the users point of view, and the most relevant to build trust among users and boost the adoption of these systems, is a perfectly tailored job for changeload-based benchmarks. In this specific case of application, we advocate the use a black-box approach, which ignores the subtleties of how the system works, and just evaluates it for the outcome that is expected of it. With such an approach, the applicability of the benchmark is not impacted by how autonomic the system is, or which parts or components of the system have self-* characteristics, as we do not need knowledge of how the system is trying (or not) to deal with the changeload during the benchmark. In fact, what we need is a way to assess how well the system deals with possible (and representative) varying scenarios it may face during its operation from the service point-of-view: we pursue a simple and practical way to assess how effective and efficient the system is in "absorbing" the perturbations and maintain (or return to) the best possible fulfillment of its goals, considering the (internal and external) context at the time, by evaluating the service provided from the user's point of view. We can, then, compare the resiliency of any system in the spectrum from *static* to completely *self-managed*.

This approach, although promising, also presents new challenges. For instance, specific goals of the compared systems may differ (e.g., different values for maximum response time, or minimum availability) or have distinct priorities. This requires benchmark metrics to realistically reflect each system's degree of achievement, by combining not only the impact of the (eventual) adaptation processes in service the delivered, but also the *weighted* evaluation of how close to their goals they were able to perform.

V. CONCLUSIONS

Considering new computing scenarios that include self-adaptive and self-organizing systems, we argue that existing benchmarking approaches have to evolve, in order to reaffirm benchmarking as a reliable source of information on the expected behavior of the systems in representative execution conditions. Beyond traditional performance metrics and workloads (which will still be needed), we claim that benchmarks must now include new features that take into account the changing nature of todays systems and runtime environments, namely changeloads, and service and adaptivity related metrics. In practice, these should focus on evaluating and comparing the ability of systems to perform while facing changeloads in their representative runtime scenarios. We believe that the same focus and effort dedicated to research in workloads characterization and modeling must be devoted to research on topics related to changeloads, as these are indeed a fundamental piece on the SASO systems benchmarking puzzle.

REFERENCES

[1] Y. Brun, "Improving impact of self-adaptation and self-management research through evaluation methodology," in *Proc. of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS'10)*, 2010, pp. 1-9.

[2] J.-C. Laprie, "From Dependability to Resilience," in *Proc. of the IEEE International Conference on Dependable Systems and Networks (DSN'08), Supplemental volume*, 2008, p. G8-G9.

[3] J. A. McCann, R. de Lemos, M. Huebscher, O. F. Rana, and A. Wombacher, "Can Self-managed systems be trusted? Some views and trends," *The Knowledge Engineering Review*, vol. 21, no. 03, p. 239, Oct. 2006.

[4] A. G. Ganek and T. A. Corbi, "The dawning of the autonomic computing era," *IBM Systems Journal*, vol. 42, no. 1, pp. 5-18, 2003.

[5] D. Garlan, "Software engineering in an uncertain world," in *Proceedings of the FSE/SDP workshop on Future of software engineering research - FoSER '10*, 2010, p. 125.

[6] M. Calzarossa, L. Massari, and D. Tessera, "Workload Characterization Issues and Methodologies," in *Performance Evaluation: Origins and Directions*, G. Haring, C. Lindemann, and M. Reiser, Eds. Springer-Verlag London, UK, 2000, pp. 459-481.

[7] P. Martin, S. Elnaffar, and T. Wasserman, "Workload Models for Autonomic Database Management Systems," in *International Conference on Autonomic and Autonomous Systems (ICAS'06)*, 2006.

[8] P. Bodik, A. Fox, M. J. Franklin, M. I. Jordan, and D. A. Patterson, "Characterizing, modeling, and generating workload spikes for stateful services," in *Proc. of the 1st ACM symposium on Cloud computing - SoCC'10*, 2010, p. 241.

[9] R. Almeida and M. Vieira, "Changeloads for Resilience Benchmarking of Self-Adaptive Systems: A Risk-Based Approach," in *Proc. of the 9th European Dependable Computing Conference (EDCC'12)*, 2012.

[10] R. Almeida and M. Vieira, "Benchmarking the resilience of self-adaptive software systems," in *Proc. of the 6th international symposium on Software engineering for adaptive and self-managing systems (SEAMS'11)*, 2011, p. 190.