Improving Impact of Self-Adaptation and Self-Management Research through Evaluation Methodology

Position Paper

Yuriy Brun
Computer Science & Engineering
University of Washington
Seattle, WA 98195-2350, USA
brun@cs.washington.edu

ABSTRACT

Today, self-adaptation and self-management approaches to software engineering are viewed as specialized techniques and reach a somewhat limited community. In this paper, I overview the current state and expectation of self-adaptation and self-management impact in industry and in premier publication venues and identify what we, as a community, may do to improve such impact.

In particular, I find that common evaluation methodologies make it relatively simple for self-adaptation and self-management research to be compared to other such research, but not to more-traditional software engineering research. I argue that extending the evaluation to include comparisons to traditional software engineering techniques may improve a reader's ability to judge the contribution of the research and increase its impact. Finally, I propose a set of evaluation guidelines that may ease the promotion of self-adaptation and self-management as mainstream software engineering techniques.

1. INTRODUCTION

Self-adaptation and self-management are two classes of techniques that may significantly improve software systems of the future. Such techniques can help tackle increased complexity of the systems themselves and of their environments. System complexity typically grows with increase in the number (and complexity) of features and requirements. Environment complexity typically grows with the size of the environment and as systems are asked to tolerate potentially faulty components, malicious attackers, and rapidly changing, unreliable resources.

Studies of self-adaptation and self-management software have become popular, with several venues emerging, e.g., the *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, the International Conference on Autonomic Computing and Communications (ICAC), the IEEE International Conferences on Self-Adaptive and Self-Organizing Systems (SASO), the Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), and the Schloss Dagstuhl seminar 08031: Software Engineering for Self-Adaptive Systems, to name just a few. Our community strongly believes that, in the near future, at least certain classes of complex software systems will rely heavily on self-adaptation and self-management [13, 20]. Despite our conviction that this work is relevant to the broad field of software engineering, with few exceptions (e.g., Calinescu and Kwiatkowska [8], Denaro et al. [15], and Dolev and Yagel [16]), our community publishes at the specialized venues, and not at the highly reputable software engineering venues (e.g., the ACM/IEEE International Conference on Software Engineering (ICSE), the European Software Engineering Conference (ESEC) and the ACM Symposium on the Foundations of Software Engineering (FSE), *IEEE Transactions on Software Engineering (TSE)*, etc.). Even conferences seemingly well-suited for self-management and self-adaptation, such as the IEEE/ACM International Conference on Automated Software Engineering (ASE), focus far more on techniques that automate software development than ones that automate runtime management.

The main goals of this paper are (1) to identify whether there exists an *impact gap*: a gap between the expected impact of self-adaptation and self-management research on software and the perceived impact of such research in the premier software engineering publication venues, and (2) to identify possible reasons for such an impact gap and develop suggestions for closing the gap.

I will first argue that the increase in the volume of research, growing system complexity, and considerable commitments from industry indicate the importance of self-adaptation and self-management techniques to software engineering. I will then quantitatively observe that such research is disproportionately underrepresented at the premier software engineering publication venues. Next, I will speculate that one way to increase such representation is by easing the comparison between self-adaptation and self-management techniques and traditional software engineering techniques. Finally, I will suggest guidelines that researchers may follow in evaluating their research to facilitate such a comparison and to more effectively demonstrate the benefits, and costs,

SEAMS '10, May 2–8, 2010, Cape Town, South Africa

of self-adaptation and self-management. It is important to note that my suggestions will not be universally applicable and, even when applicable, may not align perfectly with all researchers' goals and constraints. These guidelines can help ease certain types of comparison and should be used when such comparison is beneficial.

2. EVIDENCE OF AN IMPACT GAP

In this section, I first argue that self-adaptation and self-management software research is perceived as integral to the advance of software engineering. Toward that goal, I observe the vast growth in such research and evaluate the relevant industrial investment and commitment. I next argue that, in an apparent contradiction to my first finding and in evidence of an impact gap, self-adaptation and self-management software research is underrepresented at the premier software engineering publication venues.

2.1 Research Growth and Industrial Commitment

The emergence and growth of self-adaptation and self-management research are perhaps most evident in the number and quality of journals and professional meetings, such as conferences, that have appeared over the last few years. Among these are: the International Conference on Autonomic Computing and Communications (ICAC), started in 2004; the *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, started in 2006; the IEEE International Conferences on Self-Adaptive and Self-Organizing Systems (SASO), started in 2007; the Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), started in 2006 and grown to become a symposium starting in 2011; and the Schloss Dagstuhl seminar 08031: Software Engineering for Self-Adaptive Systems, held in 2008, as well as the upcoming sequel Schloss Dagstuhl seminar 10431, to be held in 2010. The emergence of these venues indicates not only high interest and level of productivity from the academic community, but also the recognition by the sponsoring organizations of the importance and potential impact of such research.

Scientists from distinct disciplines in software engineering have identified that self-adaptation or self-management techniques will become a necessity for software in the near future [13]. In late 2001, IBM identified the "software complexity crisis" as the main obstacle to progress in the IT industry. In particular, IBM pointed out that installation, configuration, tuning, and maintenance of modern complex software requires skilled IT professionals [23]. In IBM's view, the growing complexity of software and number and distribution of resources running such software is unsustainable, given today's approaches. The only viable option is autonomic computing, which consists of "systems that manage themselves according to an administrator's goals and components that integrate themselves as effortlessly as a new cell establishes itself in the human body" [20].

Thus, IBM has established the Autonomic Computing initiative, with the vision of developing self-managing software systems that require minimal, if any, human interaction for set up and maintenance. Toward solving what IBM calls "this vitally important problem," Kephart has outlined a number of challenges he envisions must be solved to allow autonomic computing. These challenges include not only developing autonomic systems, but also (1) high-level architectures and design patterns for autonomic elements that can be composed to create autonomic systems, (2) tools that support development of autonomic systems, and (3) models of human-computer interface interaction for proper inclusion of the users in the software control loop [22].

IBM has also been integral in promoting autonomic computing education, putting together two courses, taught at the Georgia Institute of Technology, the Ohio State University, and the North Carolina State University, and parts of which are currently being used at over 200 universities worldwide. IBM contributes financially toward the goal of promoting education and research in autonomic computing with several faculty research awards, gold sponsorship of the International Conference on Autonomic Computing and Communications ($15K annually), and sponsorship of the NSF Center for Autonomic Computing ($35K annually), combing the efforts of the University of Florida, the University of Arizona, and Rutgers, the State University of New Jersey, as well as industrial partners BAE Systems, Ball Aerospace, Intel, Microsoft, Northrop-Grumman, NEC, Raytheon, Xerox, Citrix, Imaginestics, and ISCA Technologies [24].

From the industrial production perspective, Brent A. Miller, a lead architect at IBM, considers aspects of autonomic computing to have ingrained themselves into products across IBM's entire product line. Rather than becoming a product of its own, autonomic computing improves the quality of existing, and newly developed products [24].

Google has also expressed commitment to self-adaptation and self-management, joining IBM in 2007 in the Large-Scale Internet Computing Initiative, with one of its foci on adaptive systems [28]. Their combined ongoing efforts and commitment of over $5 million in 2009 to study cloud computing, again with one of the foci on adaptive systems [29], indicate the industry's perceived future impact of self-adaptation and self-management on software.

The vision of significance of self-adaptation and self-management research is not limited to the industry. The National Science Foundation recently announced its 2010 appropriations and has specifically identified "adaptive systems technology" as one of six areas of research for which high levels of funding must be maintained, along with climate change, cyber-enabled discovery and innovation, and the National Radio Astronomy Observatory, among others [21].

It is important to point out that past experience indicates that expectations set by industry and government funding agencies may, at times, be misleading. It is not atypical for a new technology to generate a funding "boom" followed by a "bust." For example, in the late 1980s, computer-aided software engineering generated tremendous investments from industry, which largely subsided by the 1990s [11], leading to a smaller, more constant stream or funding and research. While, in this paper, I do not aim to address the question of whether the expected impact of self-adaptation or self-management software research is warranted, I do attempt to demonstrate a gap between the expected impact and the current impact in software engineering venues.

2.2 Academic Publications

I surveyed seven instances of premier conferences in software engineering: the ACM/IEEE International Conference on Software Engineering (ICSE) 2010, the joint European Software Engineering Conference (ESEC) and ACM SIG-

Venue	# of SASM papers	total # of papers	SASM ratio	SASM papers
ICSE 2010	0	52	0.0%	—
ESEC/FSE 2009	2	37	5.4%	[10, 15]
ASE 2009	2	38	5.3%	[7, 35]
ICSE 2009	5	50	10.0%	[8, 12, 17, 27, 36]
TSE 2009	0	50	0.0%	—
TOSEM 2009	1	13	7.7%	[26]
FSE 2008	2	31	6.5%	[30, 32]
ASE 2008	1	34	2.9%	[37]
ICSE 2008	1	55	1.8%	[34]
TSE 2008	4	51	7.8%	[9, 14, 16, 19]
TOSEM 2008	0	20	0.0%	—
Conference Total	13	297	**4.4%**	
Journal Total	5	134	**3.7%**	
Total	18	431	**4.2%**	

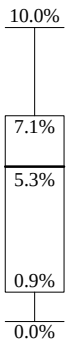

Figure 1: Less than 4.2% of all paper published in premier software engineering venues in the last two years deal with self-adaptation and self-management (SASM) research. The graphic on the right shows the minimum, 1st quartile, median, 3rd quartile, and maximum for the SASM papers distribution accross the venues.

SOFT Symposium on the Foundations of Software Engineering (FSE) 2009, the IEEE/ACM International Conference on Automated Software Engineering (ASE) 2009, ICSE 2009, FSE 2008, ASE 2008, and ICSE 2008. Further, I surveyed the past two years (2008 and 2009) of all publications in two premier software engineering journals: the *IEEE Transactions on Software Engineering* (*TSE*) and the *ACM Transactions on Software Engineering and Methodology* (*TOSEM*).

Figure 1 depicts my estimates of how many, and what fraction of the papers published at the premier software engineering venues describe self-adaptation or self-management techniques. Despite the high expectation for such techniques, outlined in Section 2.1, less than 4.2% of papers published at the premier software engineering venues over the last two years are concerned either with self-adaptation or self-management.

I used the following methodology in generating data for Figure 1. For every venue, I evaluated only the full research-track papers published at that venue. I did not evaluate short papers or papers published at special tracks at conferences. I did, however, treat special issues of journals identically to regular issues. For each paper, I considered whether its title warranted any chance that the paper dealt with self-adaptation or self-management research. Given the smallest of such possibilities, I carefully reviewed the paper's abstract and meta data (category, terms, and keywords). For most of the papers, I was at this point able to make the classification as to whether the paper described a self-adaptation or self-management technique. In a few rare cases, I was still unsure and proceeded to read the paper itself to make the classification. For a substantial number of the evaluated papers, I possessed additional information beforehand, such as having read the paper, seen a talk on the paper, or conversed with the authors of the paper about their research. Where available, I leveraged that additional information in making the classifications.

While I made every effort to be consistent and ensure that my classification methodology is repeatable, a portion of my assessment is subjective. The goal of this data is not to facilitate deep statistical analysis, but rather to demonstrate a general trend in these venues' recognition of self-adaptation and self-management research. While there are some outliers, (e.g., 10.0% for ICSE 2009), most of the values are close to the average of 4.2%. The middle half of the venues contained between 0.9% and 7.1% of papers that dealt with self-adaptation and self-management issues.

In order to fully appreciate the ratios shown in Figure 1, one could conduct a number of control experiments. For example, to ensure that my methodology for identifying self-adaptation and self-management research is accurate, I applied that methodology to papers from two specialized venues: the 2009 International Conference on Autonomic Computing and Communications (ICAC09) and the 2009 IEEE International Conferences on Self-Adaptive and Self-Organizing Systems (SASO09). I found that my methodology identified 14 of the 15 ICAC papers (93.3%), and all 27 of the SASO papers (100%) as self-adaptation and self-management research.

One other relevant control experiment that I did not perform in this study would measure how well other topics are represented at the premier software engineering venues. It may be the case that due to the diversity of software engineering research topics, 4.2% does not indicate an under-representation. However, it is unclear how to properly measure the relationship of other topics to self-adaptation and self-management, because, in some sense, self-adaptation and self-management are independant of the other topics. That is, research could cover self-adapting testing or self-managing software architecture.

It is further important to note that no conclusions about these venues' acceptance rates of self-adaptation and self-management papers can be made from the data in Figure 1. Answering that question requires data on the fraction of all submissions, as opposed to accepted submissions, that deal with self-adaptation and self-management. Unfortunately,

such data is unavailable. Nevertheless, the data presented does clearly illustrate the relatively low rate of publication of self-adaptation and self-management research in premier software engineering venues.

3. REASONS FOR THE IMPACT GAP

It is not uncommon for young fields to be confined to specialized venues until they reach an adequately mature state. While I make no claims about the positive and negative effects of such confinement, I wish to systematically explore how we can advance the field's maturity and facilitate incorporation at the premier software engineering venues.

There are likely multiple reasons for the impact gap in software engineering. These include, but are not limited to, (1) the relative immaturity of the self-adaptation and self-management software field; (2) the emergence of new venues outside of software engineering that are relevant to self-adaptation and self-management; and (3) importance of self-adaptation and self-management to artificial intelligence, pervasive and ubiquitous computing, middleware infrastructure, multiagent, and distributed robotics research, as well as other fields.

Although some of the above-outlined reasons indicate that the impact gap exists because self-adaptation and self-management research is steered toward other venues, I believe that there is significant value to software engineering in self-adaptation and self-management research being properly evaluated, reviewed, and published in software engineering venues. Further, while specialized venues exist for most software engineering topics, e.g., the International Symposium on Software Testing and Analysis for testing and analysis research, that does not prevent those topics from being well represented at the premier software engineering venues. Thus, I am concerned with the reasons self-adaptation and self-management research is not often accepted at software engineering venues, partially causing the impact gap.

In my experience as a peer-reviewer for some premier software engineering venues and some self-adaptation- and self-management-specific venues, as well as a consumer of published software engineering research, I have observed several differences in the evaluation methodologies of self-adaptation and self-management research and other software engineering research. These differences may be justified, however, they, at times, make it difficult to compare self-adaptation and self-management techniques to traditional software engineering techniques. The diversity of self-adaptation and self-management techniques often causes researchers to develop one-off evaluation methodologies. While it is certainly possible to develop a very strong one-off methodology, it is important to facilitate proper comparison between the evaluated techniques and prior art.

The following example illustrates how the evaluation methodology may make it difficult to compare a technique in question with certain existing techniques. Some self-adaptation and self-management literature is structured according to the following approach. The authors consider a problem within some dynamic environment and find that the state-of-the-art does not address some aspect of the environment's dynamics. The authors then develop a solution that addresses the previously missing dynamics and deploy that solution in the target environment, evaluating how well (quickly, reliably, consistently, etc.) the solution handles the dynamics of that environment. Finally, the authors conclude that their solution solves the problem. This approach does, in fact, present a problem, demonstrate a solution, and evaluate the solution's performance. However, a shortcoming of this approach is that, by construction, it provides no insight into the cost-to-benefit relationship of using the proposed solution. The previous state-of-the-art technique is likely to still work in the new environment, though perhaps not as well as before. And the proposed solution is likely to experience some overhead in self-adjusting to the new environment. Further, sometimes straightforward combinations of existing techniques may allow the previous state-of-the-art to handle the new dynamics. *Without a fair head-to-head comparison to existing software systems, self-adaptation and self-management software research suffers the inability to accurately present its benefits.* This methodology may be acceptable in venues specific to self-adaptation and self-management because of the environments considered typical in such research. However, in software engineering research, highly dynamic and unpredictable environments are a growing consideration, but not yet the status quo. Therefore, without a fair head-to-head comparison, it is harder for reviewers of premier software engineering venues to assess self-adaptation and self-management research than other research.

One important counterpoint to my argument for increasing the evaluation and facilitating the comparison is that research, and the papers that describe that research, have a number of constraints. Perhaps most notably, researchers have finite time and papers have limited space, making it difficult to perform all evaluation that might be desirable, and even to describe all the performed evaluation in a particular paper. Thus, the benefits of a head-to-head comparison may, for some research, be insufficient to offset its cost. It is up to individual researchers to consider how the comparison may improve, or detract, from the description of their work.

In order to describe, in greater depth, the ways in which evaluation methodology of self-adaptation and self-management literature stops short of comparison with the state-of-the-art, I will leverage four example papers on self-adaptation and self-management techniques. Two of these are my papers, published at workshops, describing two aspects of a self-organizing computational grid. The evaluation strategies in these papers have several flaws. The other two papers are award-winning premier conference publications. I have selected the latter two papers with strong evaluation methodologies purposefully, because they are clearly some of the best examples of strong publications, each winning best paper awards at their respective conferences. Nevertheless, I will attempt to identify their evaluations' strengths and weaknesses. The following is a summary of the four subject papers:

- In "An architectural style for solving computationally intensive problems on large networks," [4], a coauthor and I describe a nature-inspired architectural style, the tile style, that can distribute computation onto an untrusted network without disclosing the private data that are part of the computation.
- In "Fault and adversary tolerance as an emergent property of distributed systems' software architectures," [5], a coauthor and I describe an aspect of the tile style that allows it to self-repair and recover from faulty nodes and malicious attacks on the underlying network.

- In "Applying genetic algorithms to decision making in autonomic computing systems," [31], Ramirez et al. describe a machine-learning approach to automatically adapt remote data mirroring, at runtime, to improve data reliability and performance, while minimizing various costs. This paper received the 2009 best paper award at the 6th International Conference on Autonomic Computing and Communications.

- In "Automatically finding patches using genetic programming," [36], Weimer et al. describe an automated way to fix errors in code. While they did not design their technique to perform at runtime, it can be imagined that an extension of this technique would allow for systems to self-diagnose and self-repair a particular class of errors at runtime. This paper received the 2009 IFIP TC2 Manfred Paul Award for Excellence in Software: Theory and Practice and the ACM SIGSOFT Distinguished Papers Award at the ACM/IEEE 31st International Conference on Software Engineering.

While these subject papers were clearly successful in their respective venues, they still had some important differences in their evaluation methodologies from typical software engineering literature. While these differences were insufficient to derail these particular papers, I still consider how these papers could have eased comparison of their techniques to traditional software engineering techniques. In particular, I have identified six examples of evaluation-methodology properties that make the comparison difficult.

1. Some literature states early that current technologies cannot handle some aspect of a dynamic environment (e.g., resources being added at runtime, resource failure, noisy conditions) and never again compare the proposed approach to existing technologies. In most cases, existing technologies will still perform in the dynamic environment, though perhaps suboptimally. It is entirely possible that the overhead of self-adaptation causes a system that can handle the addition of resources at runtime to underperform a static system that simply ignores new resources. The fault-tolerance tile style paper [5] suffers from this evaluation shortcoming. The paper compares its technique to one that uses no fault tolerance, both in the benefit and the cost measures. However, it does not analyze how often failures might occur, thus making it impossible to relate the benefits to the costs.

2. Some papers claim, and perhaps argue, that some aspect of the proposed solution (e.g., decentralization) is critical to solving the target problem in certain environments and never experimentally compare the proposed solution to ones without that aspect (e.g., centralized solutions). Centralized solutions may experience single points of failure and bottlenecks, but those shortcomings will only sometimes affect the system's performance. Explicit comparison of system performance in target scenarios can demonstrate that, in fact, these shortcomings justify the overhead of self-adaptation and self-management. In the data mirroring paper [31], the authors evaluate how quickly `Plato`, their genetic algorithm-based decision-making process for finding efficient overlay networks, converges to a near-optimal solution, how close the solution is to the optimal one, how long a system takes to recover from a reconfiguration, and how much data could potentially be lost during a reconfiguration. However, they do not examine the performance of the non-adaptive system and do not demonstrate the cost of not performing a reconfiguration. Further, they never analyze how frequently the need for reconfiguration arises in real-world scenarios. In fact, in this case, `Plato` most likely provides significant improvement over the state-of-the-art; however, in order to recognize this, a reader must buy into the claim that self-adaptation is important, rather than observe actual data and make an informed judgment.

3. It is surprisingly common to compare the proposed technique to a naïve self-adapting or self-managing solution, rather than the non-self-adaptive and non-self-managing state-of-the-art. The existing solutions may have shortcomings, but without experimental comparison, the extent of those shortcomings, and the trade-off between costs and benefits of the proposed technique are unclear. Of the subject papers, the most striking example of this phenomenon is in the fault-tolerance tile style paper [5]. This paper compares its technique with the non-fault-tolerant version of itself, but never with other fault-tolerance techniques designed for other dynamic, or even static environments.

   This particular example has previously appeared in other fields of research. During the early years of research on parallel algorithms, it was quite common to compare executing an algorithm on n processors to executing that same algorithm on a single processor. In later years, the community recognized that the proper comparison is, instead, to the state-of-the-art sequential algorithms, allowing readers to judge the actual acceleration of using n processors.

4. Quite typically, self-adaptation and self-management literature does not perform formal theoretical complexity analysis of its algorithms, often due to the algorithms' distributed nature. The fact that an algorithm involves multiple, perhaps many, agents that may act in a nondeterministic or probabilistic manner, does not imply that one cannot bound the running-time and space complexities of that algorithm. In fact, the large (asymptotically approaching infinite) number of agents often makes nondeterministic and probabilistic analyses simpler. A theoretical analysis can simplify head-to-head comparisons with other techniques. Both the self-repairing data-mirroring [31] and the automated error-fixing [36] papers lack theoretical analysis, even though both would benefit from it. One important measure of the data-mirroring technique is how quickly it converges on a near-optimal solution after a perturbation in the environment. While the paper describes a number of empirical experiments demonstrating this convergence time, and a set of parameters that achieve fast convergence, the evaluation would benefit from a formal convergence analysis, as is common in machine-learning literature. Not only would such analysis bound waiting times, it may reveal the conditions necessary for convergence, and thus identify scenarios

the technique may not apply to. Similarly, the error-fixing paper lacks a formal characterization of an error and the theoretical analysis of what types of errors the technique can fix. A formal analysis could help answer two of the most important open questions of the paper: "How scalable is the technique?" and "What classes of errors can the technique fix?"

5. Some research compares the proposed technique to existing self-adaptive and self-managing solutions, assuming they are the state-of-the-art because they handle more-dynamic environments than non-self-adaptive and non-self-managing solutions. As long as one compares techniques aimed at different environments, it is unsafe to make the assumption, without evidence, that techniques aimed at similar environments will outperform techniques aimed at dissimilar ones. The data-privacy tile style paper [4] compares its approach to others that aim to preserve privacy. However, it lacks the analysis of how difficult it may be to recover private data from techniques that do not explicitly target privacy preservation, while offering other benefits. Without such an analysis, a reader cannot assess the benefit delivered by the technique over the state-of-the-art.

6. An important aspect of experimentation is repeatability. Software engineering researchers, including ones who work on self-adaptation and self-management, often make publicly available the various relevant prototype systems used for evaluation. However, quite often, for self-adaptation and self-management research, simply making the software available is not enough. In distributed, asynchronous, nondeterministic, and probabilistic environments, repeating experiments may often require recording of additional information (such as event sequences, environmental changes, and seeds for random number generation) and implementing additional control over the system (such as replaying the events and the environmental changes). None of the four subject papers [4, 5, 31, 36] facilitated experiment repeatability in such a manner.

7. The final example requires more effort to avoid than the previous ones have. It is typical to compare the proposed technique to the state-of-the-practice solution published in a paper, as opposed to the state-of-the-art solution that results from combining multiple existing techniques. For example, centralized solutions can often overcome single points of failure with uses of well-known replication and redundancy techniques. Of course, developing a combination of existing techniques may, in itself, involve substantial work and serve as a significant contribution. In such cases, it may be understandable and acceptable to omit the comparison to a not-yet-existing solution. On the other hand, in other cases, the combinations of existing techniques are borderline trivial, and comparisons to such combinations should be made.

4. EVALUATION SUGGESTIONS

My central recommendation for evaluation methodology of self-adaptation and self-management research is to include a comparison of the proposed techniques to the state-of-the-art, keeping in mind that different solutions may represent the state-of-the-art with respect to different evaluation dimensions and that state-of-the-art solutions may or may not come from other self-adaption and self-management research. Of course, it could be noted that some of this advice is applicable to and could improve not only self-adaptation and self-management research but also other software engineering research.

In this section, I go into greater depth on how one may improve the comparison with the state-of-the-art for five specific types of target scenarios self-adaptation and self-management software research addresses. It is important to note that not all of my suggestions will be suitable, or even applicable, to all techniques. It is even conceivable that employing them may hurt some evaluation methodologies. The goal of these suggestions is to allow for better comparison between the proposed techniques and the state-of-the-art, which may or may not be the aim of any particular paper.

4.1 Decentralization

Centralized systems may have single points of failure and may contain bottlenecks that adversely affect system performance. Decentralization is often a viable solution to avoiding those problems, but in many scenarios may require the individual components to have significantly more logic than if they were controlled from a central entity. As a result, decentralized systems may outscale centralized ones, but may underperform in smaller-scale environments.

When proposing a decentralized solution to a problem with existing centralized solutions, the evaluation should compare the proposed solution to the existing ones. When the target environment renders the existing solutions virtually useless, (e.g., if the aim of the technique is to scale to environments orders of magnitude larger and more error-prone than the state-of-the-art can handle), the paper should not only argue that the existing solutions cannot handle the environment, but also how likely such an environment is to exist. In other words, it should be clear to the reader that the solution not only solves a previously unsolved problem, but also that the unsolved problem is substantial.

Further, when decentralization aims to allow the system to handle a particular aspect of the environment, the evaluation should consider the result of applying any of a number of existing techniques to the existing state-of-the-art solutions. For example, if the proposed solution handles error-prone and unreliable environments, the evaluation should consider how existing techniques that improve the reliability of components (e.g., redundancy [25], self-configuring optimistic programming [1, 2], active replication [18], primary backup [6]) would affect the existing solutions. If the decentralized solution aims to remove the single-point-of-failure, the evaluation should compare that solution to the centralized solution employing some or all of these techniques to improve the reliability of its single-points-of-failure. Similarly, if the decentralized solution aims to remove bottlenecks, it must be compared to centralized solutions that employ load-balancing and channel- and component-replication techniques.

4.2 Faulty Resources

Perhaps one of the most researched areas of self-adaptation and self-management is fault tolerance: robustness to component and resource failure. In fact, the central aspect of IBM's autonomic computing is automated detection

and repair of failures at runtime. It is quite typical for self-adaptation and self-management literature to describe biology-inspired solutions that use large swarms of cheap faulty components to accomplish complex tasks. For example, some biological systems that have been used for such inspiration are termites building structurally and functionally complex mounds and bees self-organizing to optimize nectar collection.

When proposing such solutions, the evaluation should consider not only biology-inspired fault tolerance, but also other well-known fault-tolerance mechanisms. In particular, engineers have dealt with faults for a long time and have established a number of techniques for improving system reliability (e.g., as previously mentioned, redundancy [25], self-configuring optimistic programming [1, 2], active replication [18], primary backup [6]). The evaluation of a biology-inspired technique should consider whether a state-of-the-art solution, augmented with one of these well-known fault-tolerance mechanisms, might not outperform the technique in question.

4.3 Signal Noise

One common challenge tackled by self-adaptation and self-management research is runtime adjustment to noisy signals. Signal noise can come from a variety of sources, such as real-world sensors, hardware component aging, environmental changes, and availability of inaccurate models of the environment during design time.

Signal processing research has long studied noisy channels and their ability to perform their jobs of data transmission while coping with noise [33]. Similarly, artificial intelligence techniques have long dealt with sensor and channel noise [3]. When proposing solutions dealing with signal noise, the evaluation should consider how the proposed technique compares with the state-of-the-art techniques enhanced with the known mechanisms from this literature.

Further, because both noise and performance when faced with noise can be elusive concepts, a formal definition of the noise model and a theoretical evaluation of how the noise affects the performance of the technique can greatly simplify the comparison of the proposed technique and the state-of-the-art.

4.4 Management Automation

It is typically easy to make the argument that removing a human from the system-management loop reduces costs, improves speed, and perhaps reduces errors. In fact, automation has been synonymous with progress and the future ever since the 18^{th} century and the industrial revolution.

Nevertheless, self-management research can benefit from making a clear comparison between the advantages and disadvantages of removing the human from the loop. While speed, response time, and repeatability likely improve with automation, certain factors may suffer. These factors include environmental changes unexpected at design time that may fall outside of the abilities of the self-management component and detection of certain patterns within failure occurrences that may lead to unscheduled preventative maintenance and reduced catastrophic failures and down time.

When proposing solutions that automate previously human-controlled management, the evaluation should compare the proposed solutions to the human-controlled state-of-the-art. Claims about cost reduction and speed improvement must be carefully argued, as humans may, among other actions, be able to catch more distinct types of failures, allow systems to fail in less catastrophic ways, and gain valuable insight into repairing systems after crashes by observing the crashes first-hand. Users may also gain deep understanding of the system and be able to identify new requirements for future versions that may otherwise go unrecognized and unimplemented.

4.5 Environment Growth

Scalability is perhaps the most general of the challenges undertaken by self-adaptation and self-management researchers. A system's self-sufficiency and independence of human-assisted maintenance can allow it to be deployed in far larger environments, and even in environments previously unreachable by systems without such properties.

Decentralization, already discussed in Section 4.1, is one common way to improve scalability, and the evaluation suggestions for comparing decentralized and centralized systems apply here as well. In particular, it is important to evaluate both, how often the centralized systems would fail in the target environments, and the costs of those failures, in terms of performance and maintenance.

Additionally, the scalability of some techniques can be improved by leveraging replication [25], and other mechanisms from reliability and decentralization literature. When proposing techniques that improve a system's scalability, the evaluation should compare the proposed technique to the state-of-the-art enhanced with these scalability-improving mechanisms.

5. CONCLUSIONS

In this position paper, I have argued that (1) there exists a gap between the impact of self-adaptation and self-management research expected by industry and apparent in premier software engineering venues, and (2) how our community can facilitate improving the impact on those venues.

While industrial leadership and government funding agencies have high expectations for self-adaptation and self-management research, and while such research has been well represented at some quality venues, premier software engineering venues have been reluctant to publish such research, with fewer than 4.2% of all papers published there over the last two years relating to self-adaptation and self-management. I have identified several reasons for this impact gap and have focused on one of these reasons: that methodologies commonly used to evaluate self-adaptation and self-management research often make it difficult to compare such research to traditional software engineering research. I have illustrated seven ways in which such comparison is made difficult and provided examples of these ways from published literature. Finally, I have outlined five suggestions for facilitating the comparison between self-adaptation and self-management techniques and traditional software engineering technqiues.

Acknowledgments

This work is sponsored in part by the National Science Foundation under Grant number CIF-288. The author wishes to thank David Notkin for helpful discussions of this work and the anonymous reviewers for suggesting control experiments and identifying additional evaluation-methodology proper-

ties that make self-adaptation and self-management research difficult to compare to other software engineering research. These suggestions and discussions helped greatly improve this paper.

6. REFERENCES

[1] Andrea Bondavalli, Silvano Chiaradonna, Felicita Di Giandomenico, and Jie Xu. An adaptive approach to achieving hardware and software fault tolerance in a distributed computing environment. *Journal of Systems Architecture*, 47(9):763–781, 2002.

[2] Andrea Bondavalli, Felicita Di Giandomenico, and Jie Xu. A cost-effective and flexible scheme for software fault tolerance. *Journal of Computer Systems Science and Engineering*, 8(4):234–244, 1993.

[3] Rodney A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6(1&2):3–15, June 1990.

[4] Yuriy Brun and Nenad Medvidovic. An architectural style for solving computationally intensive problems on large networks. In *Proceedings of Software Engineering for Adaptive and Self-Managing Systems (SEAMS07)*, Minneapolis, MN, USA, May 2007.

[5] Yuriy Brun and Nenad Medvidovic. Fault and adversary tolerance as an emergent property of distributed systems' software architectures. In *Proceedings of the 2nd International Workshop on Engineering Fault Tolerant Systems (EFTS07)*, pages 38–43, Dubrovnik, Croatia, September 2007.

[6] Navin Budhiraja, Keith Marzullo, Fred B. Schneider, and Sam Toueg. The primary-backup approach. In *Distributed Systems*, pages 199–216. ACM Press/Addison-Wesley Publishing Co., 2 edition, 1993.

[7] Jacob Burnim, Nicholas Jalbert, Christos Stergiou, and Koushik Sen. Looper: Lightweight detection of infinite loops at runtime. In *Proceedings of the 24th IEEE/ACM International Conference on Automated Software Engineering (ASE09)*, Auckland, New Zealand, 2009.

[8] Radu Calinescu and Marta Kwiatkowska. Using quantitative analysis to implement autonomic IT systems. In *Proceedings of the ACM/IEEE 31st International Conference on Software Engineering (ICSE09)*, pages 100–110, Vancouver, Canada, 2009.

[9] Carlos Canal, Pascal Poizat, and Gwen Salaün. Model-based adaptation of behavioral mismatching components. *IEEE Transactions on Software Engineering (TSE)*, 34(4):546–563, 2008.

[10] Valeria Cardellini, Emiliano Casalicchio, Vincenzo Grassi, Francesco Lo Presti, and Raffaela Mirandola. QoS-driven runtime adaptation of service oriented architectures. In *Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE09)*, pages 131–140, Amsterdam, The Netherlands, 2009.

[11] CASE. Foreword. In *Proceedings of the 7th International Workshop on Computer-Aided Software Engineering (CASE95)*, page ix, Toronto, Canada, 1995.

[12] Hervé Chang, Leonardo Mariani, and Mauro Pezze. In-field healing of integration problems with COTS components. In *Proceedings of the ACM/IEEE 31st International Conference on Software Engineering (ICSE09)*, pages 166–176, Vancouver, Canada, 2009.

[13] Betty H.C. Cheng, Rogério de Lemos, Holger Giese, Paola Inverardi, et al. Software engineering for self-adaptive systems: A research roadmap. In *Software Engineering for Self-Adaptive Systems*, volume 5525, pages 1–26. Lecture Notes in Computer Science Hot Topics, 2009.

[14] Siu-Nam Chuang and Alvin T.S. Chan. Dynamic QoS adaptation for mobile middleware. *IEEE Transactions on Software Engineering (TSE)*, 34(6):738–752, 2008.

[15] Giovanni Denaro, Mauro Pezzè, and Davide Tosi. Ensuring interoperable service-oriented systems through engineered self-healing. In *Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE09)*, pages 253–262, Amsterdam, The Netherlands, 2009.

[16] Shlomi Dolev and Reuven Yagel. Towards self-stabilizing operating systems. *IEEE Transactions on Software Engineering (TSE)*, 34(4):564–576, 2008.

[17] Ilenia Epifani, Carlo Ghezzi, Raffaela Mirandola, and Giordano Tamburrelli. Model evolution by run-time parameter adaptation. In *Proceedings of the ACM/IEEE 31st International Conference on Software Engineering (ICSE09)*, pages 111–121, Vancouver, Canada, 2009.

[18] Pascal Felber and Andre Schiper. Optimistic active replication. In *Proceedings of the 21st IEEE International Conference on Distributed Computing Systems (ICDCS01)*, pages 333–341, Phoenix, AZ, USA, 2001. IEEE Computer Society.

[19] William G.J. Halfond, Alex Orso, and Pete Manolios. WASP: Protecting web applications using positive tainting and syntax-aware evaluation. *IEEE Transactions on Software Engineering (TSE)*, 34:65–81, 2008.

[20] IBM. Autonomic computing manifesto. `http://www.research.ibm.com/autonomic/manifesto`, 2001.

[21] Richard M. Jones. Fiscal year 2010 National Science Foundation appropriation. *The AIP Bulletin of Science Policy News*, (146), 2009.

[22] Jeffrey O. Kephart. Research challenges of autonomic computing. In *Proceedings of the 27th ACM/IEEE International Conference on Software Engineering (ICSE05)*, pages 15–22, St. Louis, MO, USA, 2005.

[23] Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *IEEE Computer*, 36(1):41–50, 2003.

[24] Paul Kontogiorgis and Brent A. Miller. Interview on the state of and IBM's contribution to autonomic computing. Personal communication, January 2010.

[25] Israel Koren and C. Mani Krishna. *Fault-Tolerant Systems*. Elsvier, Inc., 2007.

[26] Marco Mamei and Franco Zambonelli. Programming pervasive and mobile computing applications: The TOTA approach. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 18(4):1–56, 2009.

[27] Brice Morin, Olivier Barais, Gregory Nain, and Jean-Marc Jézéquel. Taming dynamically adaptive systems using models and aspects. In *Proceedings of the ACM/IEEE 31st International Conference on Software Engineering (ICSE09)*, pages 122–132, Vancouver, Canada, 2009.

[28] Press release. Google and IBM announce university initiative to address Internet-scale computing challenges. `http://www.google.com/intl/en/press/pressrel/20071008_ibm_univ.html`, 2007.

[29] Press release. National Science Foundation awards millions to fourteen universities for cloud computing research. `http://www.nsf.gov/news/news_summ.jsp?cntn_id=114686&govDel=USNSF_51`, 2009.

[30] Franco Raimondi, James Skene, and Wolfgang Emmerich. Efficient online monitoring of web-service SLAs. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE08)*, pages 170–180, Atlanta, GA, USA, 2008.

[31] Andres J. Ramirez, David B. Knoester, Betty H.C. Cheng, and Philip K. McKinley. Applying genetic algorithms to decision making in autonomic computing systems. In *Proceedings of the 6th International Conference on Autonomic Computing (ICAC06)*, pages 97–106, Barcelona, Spain, 2009.

[32] Michele Sama, David S. Rosenblum, Zhimin Wang, and Sebastian Elbaum. Model-based fault detection in context-aware adaptive applications. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE08)*, pages 261–271, Atlanta, GA, USA, 2008.

[33] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, July and October 1948.

[34] Sylvain Sicard, Fabienne Boyer, and Noel De Palma. Using components for architecture-based management: the self-repair case. In *Proceedings of the ACM/IEEE 30th International Conference on Software Engineering (ICSE08)*, pages 101–110, Leipzig, Germany, 2008.

[35] Yiqiao Wang and John Mylopoulos. Self-repair through reconfiguration: A requirements engineering approach. In *Proceedings of the 24th IEEE/ACM International Conference on Automated Software Engineering (ASE09)*, Auckland, New Zealand, 2009.

[36] Westley Weimer, ThanhVu Nguyen, Claire Le Goues, and Stephanie Forrest. Automatically finding patches using genetic programming. In *Proceedings of the ACM/IEEE 31st International Conference on Software Engineering (ICSE09)*, pages 364–374, Vancouver, Canada, 2009.

[37] Andrea Zisman, George Spanoudakis, and James Dooley. A framework for dynamic service discovery. In *Proceedings of the 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE08)*, pages 158–167, L'Aquila, Italy, 2008.