ICSE 2003 WADS Panel: Fault Tolerance and Self-Healing

Rogério de Lemos
Computing Laboratory
University of Kent at Canterbury, UK

The objective of this panel was to discuss, in the context of software architectures, the differences between the existing area of fault tolerance, and the upcoming area of self-healing. The panel took a critical view on the differences between these two areas and their potential roles in the design and construction of trustworthy systems. This Panel had as panellists Valerie Issarny from INRIA – France, and Philip Koopman from CMU – USA, and as a moderator Rogério de Lemos from the University of Kent – UK.

The moderator started the Panel by bringing into light commonly used interpretations of both terms. Fault tolerance is one of the means to dependability, and it refers to the mechanisms and techniques that enable a system to deliver its specified service despite the presence of faults [Avizienis 2001]. On the other hand, since the idea of self-healing has only recently caught the imagination of several research communities within computer science, there was not yet a consensus what it should stand for. This is no exception within the community of software engineering, which has recently organised its first ACM SIGSOFT Workshop on Self-Healing Systems (WOSS'02) [Garlan 2003]. A relevant question to be raised was why this sudden interest by the software engineering community in self-healing, considering that the hardware community had the same aspirations for their systems for quite awhile. A possible answer was that hardware engineers knew from the outset that they could not build systems that would not fail, essentially due to the low reliability of their hardware components. In contrast, the worries of software engineers were not restricted with the potential failures of individual components, but with the failure of the whole systems: not only from the perspective on how to put several components together, but also on the process to achieve this. The main reason being that faults plaguing software systems were of different type, which would require more sophisticated and costly means for incorporating redundancies in order to avoid system failures. Hence the view that software engineering has been a discipline concerned with the development of software systems based on rigorous (sometimes formal) languages and processes, supported by tools, aiming to build software systems without bugs. From the dependability perspective this would entail fault prevention, in the form of rigorous designs (e.g., goal structures, and UML), fault removal, in the form of verification and validation (e.g., model checking, and testing), and fault forecasting, in the form of system evaluation (e.g., software metrics).

Based on the above considerations the moderator conjectured what would be the role of self-healing within software engineering. Two potential roles would be possible. In the first, self-healing would be associated with the capability of a software system in dealing with bugs. For example, if the interface types of two components do not match, then the resolution of such mismatch should rely on run-time solutions rather than design time. Making an analogy with nature, this would entail the healing of a wound. In the second role, self-healing would be associated with the capability of a software system in adapting to changes in its environment. For example, the performance of a system could be adjusted depending on the availability of its resources. Making an analogy with human nature, this would be related to the process of body sweating for regulating its temperature. However, in this role, the notion of "healing" is not evident, and it would be more related to the notion of software homeostasis, put forward by Mary Shaw. However, whatever the role taken, it seems that the software engineering community is moving from design time to run time solutions, perhaps to solve the same kind of problems that the hardware community faced decades ago.

Aiming to be controversial, the moderator championed fault tolerance and self-healing in terms of their achievements and potentialities, from the perspective of dependability and software engineering, respectively. From the dependability perspective, self-healing could be seen as re-inventing the wheel, since all problems and solutions being investigated have already been mapped out. In contrast, from the software engineering perspective, fault tolerance provides expensive solutions, since the provision and management of redundancies is also expensive, leading to complex systems. Moreover, although solutions might be elegant, in general they are impractical. Based on these two conflicting perspectives, the question that remains to be answered is whether there exists a middle way between them, considering that both communities are trying to handle the same kind of problems. However, it seems that the software engineering community is motivated to pursue their own solutions because the dependability "framework" might appear too strict or inappropriate for their own problems. For example, fault tolerance has been proven to be effective at the later stages of design, in contrast with architectural solutions emerging from self-healing of software system. From an historical perspective, as resources become less costly, it makes sense to place the redundancies at higher levels of abstraction, though this might lead to a wasteful of resources. The moderator concluded by identifying the lack of a coherent terminology as a major impediment for self-healing community to have a common understanding of the scope of their field. For example, there is a need to establish what are the undesirable, though not unexpected circumstances that should triggered system reaction to changes. Are these the threats to dependability, or are there other notions that should be able to capture change, whatever this might be? Again contrasting with the dependability terminology, what are the attributes of self-healing that would allow measuring the effectiveness of the means for pre-empting or mending potential disturbances to the quality of the system services?

Valerie Issarny started her talk by questioning what was self-healing about? Was it a new name for fault tolerance? Fault tolerance has for long been associated with systems fixed at design time, in which the correctness of these systems was enforced via rigorous design, in same cases, formal methods. An example of such systems was safety-critical systems, which are usually based on a dedicated design, software and hardware. However, the features of today's systems, such as increasingly consumer-oriented and highly dynamic (evolving user requirements, evolving environment, quality of services, and mobility), would be difficult to support if these more conventional and rigid approaches were used. The reason being that it is expected these systems to deliver the advertised functionality, in all situations, possibly with different levels of quality of services. Based on these expectations, Valerie Issarny conjectured that this would be sufficient to justify the launching of a new area of research on self-healing system that would deal with fault tolerance for dynamic systems. Evidence supporting this claim, she said, can be found on the different interpretations given to the field during WOSS'02. For example, Mary Shaw mentions that self-healing deals with imprecise specification and uncontrolled environment, while other statements show an almost consensus that self-healing deals with the adaptation and/or reconfiguration of systems according to their dynamics. In her final remarks, Valerie Issarny questioned why adaptation is part of fault tolerance and self-healing? In her opinion there is the need for reconsidering the design of software systems that are dynamic in nature. This should be considered in the system's standard specification that should take into account its dynamics, the system's correctness that should be in accordance to both functional behavior and provided level of quality of services, and the specification of fault tolerance when standard specification was not met. However, this approach raises the question what should be standard or exceptional behaviour. For example, the disconnection in mobile systems, what it should be considered?

Philip Koopman began his presentation by prompting three questions that would roughly be able to identify what was the existing perception in the scopes of fault tolerance and self-healing: was *some* fault tolerance also self-healing? Was *all* fault tolerance also self-healing? Was *all* self-healing also fault tolerance? The brief answers to these questions were respectively: yes, no, and maybe (or assume yes until proven otherwise). But before he could proceed in substantiating the arguments of the above three questions, Philip Koopman questioned whether it was the right thing to compare fault tolerance and self-healing. Fault tolerance essentially is an emergent property: systems are either fault tolerant or not, to varying degrees. It is also a measurable property, in sense that fault injection experiments could be employed to check which faults can really be tolerated. On the other hand, self-healing seemed like an approach, or point of view. For example, what was an "injury" and what was not? It seemed also there were no unifying themes to "self-healing". On the measurement aspect, were there self-healing outcomes that were fault tolerance, in the sense of dependability? In a wider sense, would one be able to measure "healability"?

Concerning the first question, whether some fault tolerance was also self-healing, Philip Koopman referred to a paper published in 1969 by Bouricius, Carter, and Schneider, entitled "Reliability Modeling Techniques for Self-repairing Computer Systems", in which the possibility of computers healing themselves had already been raised. He also mentioned that an early self-healing idea was standby sparing, where in a pool of operating units, if one broke, a spare one was used to replace it. If that was not healing, then there was the need for a tighter definition of "healing". This reasoning could also be extended to other traditional fault tolerant techniques like the Byzantine generals algorithms, and error correcting codes. On the second question, whether all fault tolerance was also self-healing, Philip Koopman mentioned that many fault tolerant techniques were probably not self-healing, and he gave some examples: the usage of highly reliable components (bullet-proof vests are not "healing"), and the employment of fail-fast or fail-silent components (component suicide is not "healing"). Concerning the latter, although such approach might not be considered "healing", it can nevertheless facilitate the healing at the system level. He also mentioned that a major characteristic of fault tolerance was to emphasize 100% functionality, and this was not clear in the context of self-healing. In conclusion, one could argue that much of fault tolerance is self-healing. On the third, and last question that he raised, whether all self-healing was also fault tolerance, Philip Koopman said that its answer was depended on how broad was the question. In a narrow sense, and taking an historical perspective, could all self-healing also be fault tolerance? Not clear, since there are a lot of issues that were not properly dealt in fault tolerance, like incomplete specification and the interaction between computers and humans. If a broader sense was taken, could all self-healing also be considered as part of fault tolerance research. The answer would be probably positive. Finally, in the broadest sense, could all self-healing also be related to dependability. Certainly yes, since the definition of dependability has become very inclusive. To conclude his talk, Philip Koopman said that the question he is concerned about was research community interactions, not its turf battles.

During discussion, after the short presentations, several other important issues were also raised. Alexander Romanovsky made comment that the split between self-healing approach from software engineering and the fault tolerant approach from dependability perhaps had its origins from the 1968 NATO conference [Naur 1969], and now it appears that both fields are converging on the understanding how to handle undesired circumstances. While software engineering has begun to consider the presence of faults, dependability wants to use software engineering based approaches to support fault tolerance. On the differences between fault tolerance and self-healing a participant made the comment that self-healing provides cost-effectiveness when dealing with problems, since one does not know during design time what the resources one might have during run time. For example, David Garlan's group focus on system dynamicity because self-healing is about run time issues. In real life there is not always absolute dependability, very often the system provides partial service, and the reason for this is that the specifications are usually not complete. In contrast, fault tolerance is about providing the specified service. However Philip Koopman put forward the idea of graceful degradation, which he said it was not self-healing. Moreover, the development of systems based on graceful degradation is expensive, but he was working on this. In terms of cost, the panel shared the opinion that self-healing might not always be cheaper, it may actually be very expensive. Another issue that was raised by the audience was how fault tolerance and self-healing were able to handle changes in the environment of the system. The understanding was that the self-healing community claims that they are able to deal with them. On the other hand, although fault tolerance cannot claim such capabilities, it is able nevertheless to provide some guarantees. These would be based on the system fault assumptions, which provide the basis for fault tolerance, however it is not clear what would be the analogous for self-healing. Nicole Levy asked how fault tolerance and self-healing dealt with hardware failures. In the panel's opinion there was no doubts that fault tolerance was able to deal with them, however it was not clear what was the self-healing approach to hardware faults. Ada Diaconescu asked whether fault tolerance was also about quality of services. The panel answered that indeed it was, but only some non-functional properties are clearly part of it. For example, timeliness - in fault tolerance one either meets a deadline or not.

As a wrapping up to the panel, the moderator reiterated the purpose of the panel, which was to have a constructive discussion on the differences and similarities between the areas of fault tolerance and self-healing, and hoped that the presentations and their subsequent discussion were fruitful enough to promote a more close collaboration between the dependability and software engineering communities.

References

[Avizienis 2001] A. Avizienis, J.-C. Laprie, and B. Randell. *Fundamental Concepts of Dependability*. Technical Report 739. Department of Computing Science. University of Newcastle upon Tyne. 2001.

[Garlan 2003] D. Garlan, J. Kramer, and A. Wolf. *Proceedings of the First ACM SIGSOFT Workshop on Self-Healing Systems (WOSS'02)*. Charleston, SC, USA. November 2002.

[Naur 1969] P. Naur, and B. Randell, (Eds.). *Software Engineering: Report of a Conference Sponsored by the NATO Science Committee*. Garmisch, Germany. October 1968. NATO Scientific Affairs Division. 1969.

[Shaw 2002] M. Shaw. ""Self-Healing": Softening Precision to Avoid Brittleness. *Proceedings of the First ACM SIGSOFT Workshop on Self-Healing Systems (WOSS'02)*. Charleston, SC, USA. November 2002. pp. 111-113.