A Requirements-Based Approach for the Design of Adaptive Systems

Vítor E. Silva Souza
Department of Information Engineering and Computer Science - University of Trento
Via Sommarive, 14 - Trento, Italy - 38123
vitorsouza@disi.unitn.it

***Abstract*—Complexity is now one of the major challenges for the IT industry [1]. Systems might become too complex to be managed by humans and, thus, will have to be self-managed: Self-configure themselves for operation, self-protect from attacks, self-heal from errors and self-tune for optimal performance [2]. (Self-)Adaptive systems evaluate their own behavior and change it when the evaluation indicates that it is not accomplishing the software's purpose or when better functionality and performance are possible [3].**

To that end, we need to monitor the behavior of the running system and compare it to an explicit formulation of requirements and domain assumptions [4]. Feedback loops (e.g., the MAPE loop [2]) constitute an architectural solution for this and, as proposed by past research [5], should be a first class citizen in the design of such systems. We advocate that adaptive systems should be designed this way from as early as Requirements Engineering and that reasoning over requirements is fundamental for run-time adaptation.

We therefore propose an approach for the design of adaptive systems based on requirements and inspired in control theory [6]. Our proposal is goal-oriented and targets software-intensive socio-technical systems [7], in an attempt to integrate control-loop approaches with decentralized agents inspired approaches [8]. Our final objective is a set of extensions to state-of-the-art goal-oriented modeling languages that allow practitioners to clearly specify the requirements of adaptive systems and a run-time framework that helps developers implement such requirements. In this 2-page abstract paper, we summarize this approach.

I. Background and Related Work

Our proposal is based on Goal-oriented Requirements Engineering (GORE), which has some of its most well-known approaches summarized in [9]. Our research, however, is not based on any specific approach, but on the main concepts that most of them present: *Goals*, *softgoals*, *quality constraints* and *domain assumptions* [10]. We assume the existence of one or more goal models whose leaf-level goals (*tasks* or *plans*) can be traced back from implemented, run-time components of the system.

Among the many architectural solutions for adaptation, our approach is based on feedback loops. The main idea of feedback control is to use measurements of a system's outputs to achieve externally specified goals (specified as reference input) [6]. A more detailed control-theoretic view of adaptive systems as feedback control systems in the context of our research is provided in a position paper [11].

Combining requirements and feedback loops invariably leads to requirements monitoring. There has been preliminary work on run-time monitoring of requirements conformance [12] and reconciliation of run-time behavior based on requirements [13]. Our proposal is based on the system being able to monitor its own requirements at runtime.

There are several different proposals for the design of adaptive systems. Many of them focus on the architectural aspects of system development, such as the dynamic planning system approach of [3], the Rainbow framework [14], the three-layered reference architecture of [15], etc. Our work differs from these by focusing on requirements.

Other proposals also focus on requirements for adaptive systems. KAOS-based proposals (e.g., [16], [17]) use linear temporal logic to specify requirements, which can be too heavy a formalism in many situations. RELAX [18] is based on SHALL statements and fuzzy branching temporal logic, which is also more complex than the approach we propose. Tropos-based proposals (e.g., [19], [20]) use the Belief-Desire-Intention model as reference architecture whereas our focus in on the feedback loop architecture. Some approaches (e.g., [21]) provide adaptivity only to changes in the environment (context), missing an important aspect which is adapting to requirements failures. By considering domain assumptions as part of the requirements, our proposal can also adapt to environment changes.

Moreover, a fundamental difference from our approach and the state-of-the-art in goal-based adaptive systems design is the fact that goals are not necessarily treated as invariants that must always be achieved. Instead, we accept the fact that the system may fail in achieving any of its initial requirements and, by considering feedback loops as first class citizens in the language, provide a way of specifying the level of criticality of each goal as constraints on their success/failure and assigning adaptation actions to be carried out when the system does not fulfill these constraints. This approach is summarized next.

II. Approach, Results and Contributions

The first step of the approach is the elicitation of Awareness Requirements (*AwReqs*), which are requirements that talk about the states assumed by other requirements — such as their success or failure — at runtime [22]. AwReqs constitute the requirements for the feedback loops that implement the adaptive capabilities of the system. In other words, they represent undesirable situations to which stakeholders would like the system to adapt, in case they happen.

For instance, in a Computer-aided Ambulance Dispatch (CAD) system [23], "goal *Register call* should never fail" (labeled *AR15*) and "quality constraint *Ambulances arrive in 8 min* should have 75% success rate" (*AR3*) are examples of *AwReqs*. Note how the former refers to single instances of requirements, whereas the latter refers to the whole requirement class in an aggregate way. *AwReqs* can also refer to other *AwReqs*, e.g., "*AR1* should succeed 90% of the time" (meta-*AwReq AR2*). At runtime, the elements of the goal model are represented as classes and instances of these classes are created every time a user starts pursuing a requirement or when they are bound to be verified. Their state (*succeeded*, *failed*, etc.) is then monitored by the infrastructure presented in [22].

AwReqs can be used as indicators of requirements convergence at runtime. If they fail, a possible adaptation strategy is to search the solution space to identify a new configuration (i.e., values for system parameters) in a way that would improve the necessary indicators. As in control systems, to know the effect each parameter change has on indicators we conduct *System Identification* for the adaptive system [24], identifying (a) parameters that can be changed at runtime — OR-refinements (variation points), already present in systems with high variability, and control variables (abstractions over large/repetitive variation points) — and (b) qualitative information about their relation to the satisfiability of *AwReqs* at runtime — using differential relations.

In the CAD system, parameter *Level of Automation* (*LoA*), with values *manual*, *semi-automatic* and *automatic*, can affect the time taken for an ambulance to arrive, thus affecting *AwReq AR3*. We represent the fact that increasing the level of automation hinders the success of *AR3* (dispatches done using a manual process are usually better) by stating the derivative of an imaginary function $AR3 = f(LoA)$ is negative: $\Delta\,(AR3/LoA) < 0$. Parameters can also be numeric (e.g., *Number of Staff Members*) and, in that case, qualitative landmark values can specify boundaries for relations when they are not applicable in the default interval $[-\infty, \infty]$.

To "close the feedback loop", in a recently submitted paper [25], we propose a framework within which failure of monitored *AwReqs* lead to new behaviors that consist of selecting new values for the system parameters. Inspired by control theoretic concepts recast in qualitative terms, the framework's controller parses the goal model, including indicators, parameters and their relations, in order to know the desired output and the space of possible behaviors for getting it. Furthermore, the framework is extensible in order to accommodate different levels of precision, which allows the adaptation mechanisms to improve along with the evolution of the model. For instance, in case of a failure in *AR3*, the framework chooses one or more parameters (e.g., *LoA*), changes their values (e.g., *automatic* $\longrightarrow$ *semi-automatic*) and re-evaluates the indicator (did the next ambulance arrive within 8 min?), iterating when the problem persists.

In some cases, however, adaptation can come not from exploring the solution space, but instead the problem space, i.e., the requirements themselves. For this reason, we propose the elicitation and modeling of *Evolution Requirements* (*EvoReqs*) [26] which consist of specific changes to be carried out on the requirements model (again, at the instance and/or class levels), under specific circumstances. *EvoReqs* are specified using a set of primitive operations to be performed over the elements of the model. Each operation is associated with application-specific actions to be implemented in the system. Furthermore, they can be combined using patterns in order to compose *adaptation strategies* (e.g., "Retry", "Delegate", "Relax", etc.).

Such patterns are then associated with specific *AwReqs* in the model and can be used at runtime by an Event-Condition-Action-based process in order to direct the system on how to adapt. This process coordinates the different applicable strategies, choosing which one to apply and checking if the problem they attempt to remedy has been solved. In the CAD system, for example, goal *Register call* is satisfied by the successful execution of tasks *Input emergency information* and *Detect caller location*. When *AR15* fails, possible adaptation strategies are to retry the goal or to relax it by disabling caller location detection.

Along with our proposals we have presented initial validation results obtained by the execution of simulations that mimic the behavior of real systems in specific failure scenarios, verifying that the adaptation framework responds appropriately. To fully evaluate the approach, however, we plan to run similar scenarios in real systems, preferably with the participation of industry partners as stakeholders. The modeling language should also be evaluated through surveys with requirements engineering practitioners.

In summary, the contributions of this research are: (a) New types of requirements (*AwReqs* [22] and *EvoReqs* [26]) that specify the requirements for a feedback control loop that implements adaptivity for a target system; (b) A systematic process for conducting *System Identification* [24] and a framework that reconfigures the system based on indicator/parameter qualitative information [25]; and (c) Tools to facilitate the design (CASE tool) and implementation (framework) of adaptive systems using this approach (source code available at github.com/vitorsouza/Zanshin). Some of these contributions are currently work-in-progress.

ACKNOWLEDGMENT

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

References

[1] P. Horn, "Autonomic Computing: IBM's Perspective on the State of Information Technology, http://www.research.ibm.com/autonomic/manifesto/," October 2001.

[2] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[3] R. Laddaga and P. Robertson, "Self Adaptive Software: A Position Paper," in *Proc. of the 2004 International Workshop on Self-* Properties in Complex Information Systems*, 2004.

[4] P. Oreizy *et al.*, "An Architecture-Based Approach to Self-Adaptive Software," *IEEE Intelligent Systems*, vol. 14, no. 3, pp. 54–62, 1999.

[5] B. H. C. Cheng *et al.*, "Software Engineering for Self-Adaptive Systems: A Research Roadmap," in *Software Engineering for Self-Adaptive Systems*, ser. Lecture Notes in Computer Science, B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, Eds. Springer, 2009, vol. 5525, pp. 1–26.

[6] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury, *Feedback Control of Computing Systems*, 1st ed. Wiley, 2004.

[7] V. Bryl, "Supporting the Design of Socio-Technical Systems by Exploring and Evaluating Design Alternatives," PhD Thesis, University of Trento, 2009.

[8] J. Andersson, R. de Lemos, S. Malek, and D. Weyns, "Modeling Dimensions of Self-Adaptive Software Systems," in *Software Engineering for Self-Adaptive Systems*, ser. Lecture Notes in Computer Science, B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, Eds. Springer, 2009, vol. 5525, pp. 27–47.

[9] A. Lapouchnian, "Goal-Oriented Requirements Engineering: An Overview of the Current Research," University of Toronto, Canada (available online: http://www.cs.toronto.edu/~alexei/pub/Lapouchnian-Depth.pdf), Tech. Rep., 2005.

[10] I. Jureta, J. Mylopoulos, and S. Faulkner, "Revisiting the Core Ontology and Problem in Requirements Engineering," in *Proc. of the 16th IEEE International Requirements Engineering Conference*. IEEE, 2008, pp. 71–80.

[11] V. E. S. Souza and J. Mylopoulos, "From Awareness Requirements to Adaptive Systems: a Control-Theoretic Approach," in *Proc. of the 2nd International Workshop on Requirements@Run.Time*. IEEE, 2011, pp. 9–15.

[12] B. H. C. Cheng and J. M. Atlee, "Research Directions in Requirements Engineering," in *Future of Software Engineering (FOSE '07)*. IEEE, 2007, pp. 285–303.

[13] M. Feather, S. Fickas, A. van Lamsweerde, and C. Ponsard, "Reconciling system requirements and runtime behavior," in *Proc. of the 9th International Workshop on Software Specification and Design*. IEEE, 1998, pp. 50–59.

[14] D. Garlan, S.-W. Cheng, A.-C. Huang, B. Schmerl, and P. Steenkiste, "Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure," *Computer*, vol. 37, no. 10, pp. 46–54, 2004.

[15] J. Kramer and J. Magee, "Self-Managed Systems: an Architectural Challenge," in *Future of Software Engineering (FOSE '07)*. IEEE, 2007, pp. 259–268.

[16] G. Brown, B. H. C. Cheng, H. Goldsby, and J. Zhang, "Goal-oriented Specification of Adaptation Requirements Engineering in Adaptive Systems," in *Proc. of the 2006 International Workshop on Self-adaptation and Self-managing Systems*. ACM, 2006, pp. 23–29.

[17] L. Baresi, L. Pasquale, and P. Spoletini, "Fuzzy Goals for Requirements-driven Adaptation," in *Proc. of the 18th IEEE International Requirements Engineering Conference*. IEEE, 2010, pp. 125–134.

[18] J. Whittle, P. Sawyer, N. Bencomo, B. H. C. Cheng, and J.-M. Bruel, "RELAX: Incorporating Uncertainty into the Specification of Self-Adaptive Systems," in *Proc. of the 17th IEEE International Requirements Engineering Conference*. IEEE, 2009, pp. 79–88.

[19] M. Morandini, L. Penserini, and A. Perini, "Towards Goal-Oriented Development of Self-Adaptive Systems," in *Proc. of the 2008 International Workshop on Software Engineering for Adaptive and Self-managing Systems*. ACM, 2008, pp. 9–16.

[20] F. Dalpiaz, P. Giorgini, and J. Mylopoulos, "Adaptive socio-technical systems: a requirements-based approach," *Requirements Engineering*, pp. 1–24, 2012.

[21] N. A. Qureshi and A. Perini, "Engineering Adaptive Requirements," in *Proc. of the 2009 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, 2009, pp. 126–131.

[22] V. E. S. Souza, A. Lapouchnian, W. N. Robinson, and J. Mylopoulos, "Awareness Requirements for Adaptive Systems," in *Proc. of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. ACM, 2011, pp. 60–69.

[23] V. E. S. Souza, "An Experiment on the Development of an Adaptive System based on the LAS-CAD," University of Trento (available at: http://disi.unitn.it/~vitorsouza/a-cad/), Tech. Rep., 2012.

[24] V. E. S. Souza, A. Lapouchnian, and J. Mylopoulos, "System Identification for Adaptive Software Systems: a Requirements Engineering Perspective," in *Conceptual Modeling – ER 2011*, ser. Lecture Notes in Computer Science, M. Jeusfeld, L. Delcambre, and T.-W. Ling, Eds. Springer, 2011, vol. 6998, pp. 346–361.

[25] ——, "Requirements-driven Qualitative Adaptation," in *submitted for publication (under review)*, 2012.

[26] ——, "(Requirement) Evolution Requirements for Adaptive Systems," in *Proc. of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (to appear)*, 2012.