Survivability-Oriented Self-Tuning of Web Systems

Bihuan Chen[1], Xin Peng[1], Yijun Yu[2], Wenyun Zhao[1]
[1]School of Computer Science, Fudan University, Shanghai 201203, China
[2]Department of Computing, The Open University, Milton Keynes, UK
{09210240005, pengxin, wyzhao}@fudan.edu.cn, y.yu@open.ac.uk

ABSTRACT
Running in a highly uncertain and changing environment, Web systems cannot always provide full set of services with optimal quality, especially when the workload is high or failures in subsystems occur frequently. It is thus desirable to continuously maintain a high satisfaction level of the system value proposition, hereafter *survivability assurance,* while relaxing/sacrificing certain quality/functional requirements that are not crucial to the survival of the Web systems. In this paper, we propose a requirements-driven self-tuning method for survivability assurance of Web systems. Using a value-based feedback controller plus a requirements-oriented reasoner, our method makes both quality and functional requirements tradeoffs decisions at runtime.

Categories and Subject Descriptors
D.2.10 [**Software Engineering**]: Design – *methodologies.*

General Terms
Algorithms, Reliability, Measurement, Performance

Keywords
Survivability, Self-Tuning, Value, Requirements, Reasoning

1. INTRODUCTION
Failing to have such a Web system that is reliably working and resilient to changes in the running environment, the business can lose their customers because it is easy for customers to transfer to competing Web sites [1]. On the other hand, the uncertainty and unpredictability of the running environment of Web systems makes it hard to constantly provide full set of services with optimal quality, especially when the workload is high or failures in subsystems occur frequently. Rather than absolute reliability, it is therefore survivability that is more practical and reasonable for Web systems. Here *survivability* is the capability of ensuring critical system services under adverse conditions, with acceptable quality degradation or even sacrifice of some desired services [2]. In this work, we show that survivability can be assured by autonomic runtime reconfigurations rather than by error-prone human intervention on the runtime structures and behaviors of the software system. Furthermore, survivability can be interpreted quantitatively to guide the reconfigurations precisely.

In order to achieve autonomic survivability assurance, we adopt the MAPE (Monitor, Analyze, Plan, Execute) control loop [3] and propose a requirements-driven self-tuning method. Extending our previous work on runtime tradeoffs about quality requirements [4], the proposed method further takes into account runtime tradeoffs about functional requirements. Quality tradeoffs concern *when* and *which* desired quality requirements may be relaxed to ensure other more critical but conflicting ones. Similarly, functional tradeoffs concern *when* and *which* desired functional services can be unbound to ensure the critical and essential services and their quality, and such unbound services can be bound to preserve the functional integrity whenever it is possible. To make such functional tradeoffs decisions, we use a reasoning algorithm that is performed on an enriched requirements goal model. Then we interpret survivability from the perspective of value-based software engineering [5] as the capability of maximizing the satisfaction level of the system value proposition, which defines how the "earned business value" at system side is measured to facilitate the "when*"* part of the problem. On the other hand, the goal models enriched with annotations of value contributions facilitate the "which*"* part of the problem.

WWW 2011, March 28–April 1, 2011, Hyderabad, India.
ACM 978-1-4503-0637-9/11/03.

2. OUR METHOD

2.1 Value-Based Survivability
We use goal modeling [6] to provide a formal representation of runtime requirements to facilitate reasoning. The goal model for an online shopping system is presented in Figure 1. Observing that some hard goals in rounded rectangles can be unbound without influencing the achievement of their parent goals, e.g., customers can partially achieve **Online Shopping** even if **Make Review** is unbound from the system. We enrich AND/OR decompositions of the goal model with the *relative parent value* to reflect the relative importance of sub-goals or their value contribution to parent goals from a business perspective. For simplicity, the relative parent value is defined as a ratio between 0.0 and 1.0. When it is 1.0, the goal must be retained to achieve the value of its parent goal; otherwise, the value of the parent goal can still be partially achieved even when this goal is unbound. The value contribution annotations are provided by business experts to deal with economic and marketing factors such as consuming behaviors of customers.

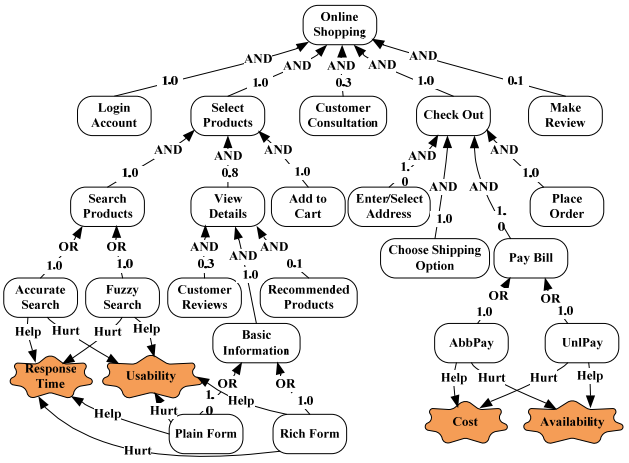

Figure 1. A goal model enriched with relative parent values.

To gauge the satisfaction level of system value proposition, it is required to measure the earned business value formally. However, many factors can influence the earned business value in a direct or indirect way. In our example, sales is a direct factor while products' details, recommended products, customers' reviews and consultations are indirect factors. Hence, a formula should be provided by a comprehensive analysis on such complicated factors by marketing experts. To illustrate the point, we simplify the formula in our example by considering the above 5 factors: each corresponds to a kind of runtime transaction that produces certain profit, e.g., a transaction of making review produces $0.032.

2.2 Self-Tuning Framework

To achieve the requirements-driven self-tuning of Web systems, our method postpones the design-time quality and functional tradeoffs decisions to the runtime, and adapts them autonomically in response to changes in the environment in order to maximize the satisfaction level of system value proposition. Figure 2 presents our self-tuning framework, illustrating the main components and their mappings to the MAPE control loop.

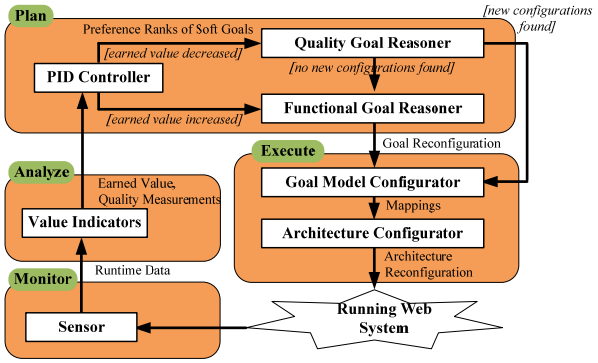

Figure 2. Our self-tuning framework.

Monitor. We use *sensors* to collect runtime data, e.g., the failure/success of a payment service, etc., through logging records;

Analyze. By analysis of the runtime data, we use *value indicators* to measure the earned business value as shown in Section 2.1, and obtain the quality measurements, e.g., availability of the payment service, etc., of related quality requirements;

Plan. Based on analyze, we use the *PID controller* to decide whether or not to make quality or functional tradeoffs decisions to maximize the satisfaction level of system value proposition. In the former case, the *PID controller* is also used to rank the preference of related soft goals to guide the reasoning for a reconfiguration to the goal model;

Execute. Finally, we use the *goal model configurator* to reconfigure the goal model according to the planned goal reconfiguration, and then we use the *architecture configurator* to execute the adaptation by reconfiguring the runtime architecture according to the mappings between the goals and the architectural components. Such architecture reconfigurations are currently supported by a reflective component model.

The **monitor** runs all the time while **analyze**, **plan** and **execute** run iteratively at regular time unit, e.g., per every minute. Furthermore, there are three possible **planning** paths.

(p1) When the earned business value decreases by a certain degree, the *quality goal reasoner* is triggered to generate a set of configurations that reach the satisfaction levels of high-ranked soft goals while relaxing the satisfaction levels of low-ranked soft goals. Here every configuration is a selection of the OR-decomposition goals.

(p2) If no new configurations are found by the *quality goal reasoner*, the *functional goal reasoner* is triggered to generate a reconfiguration that optimizes the satisfaction levels of high-ranked soft goals while sacrificing one leaf-level hard goal with the minimal relative root value defined as the product of the relative parent values along the path from root goal to the hard goal.

(p3) When the earned business value increases by a certain degree, the *functional goal reasoner* is triggered to generate a reconfiguration that improves the functional integrity of the Web system by rebinding the recently sacrificed leaf-level hard goal.

Note quality tradeoffs decisions are made to replace one component with another and the functional tradeoffs decisions are made to unbind or bind a component. Comparing with a replacement, unbinding or binding introduces more radical change to the architecture. And there is often an uncertainty in whether or not such a radical action has taken effect on the target system [7]. As a result, the system could suffer oscillations from such unbinding and binding actions (e.g., alternating the same component). Thus we adopt a timed delay for the effect of the action to be achieved in functional tuning execution as suggested by S. W. Cheng et al. [7].

3. CONCLUSIONS

With our value-based interpretation of survivability assurance, we have proposed a requirements-driven self-tuning method for survivability assurance of Web systems. The method employs both quality and functional requirements tradeoff through goal-oriented reasoning. We intend to investigate the timed delay and its impact more deeply and more precisely in the near future and to apply our method in the new paradigm of cloud computing.

4. ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China under Grant No. 90818009.

5. REFERENCES

[1] J. Offutt. Quality Attributes of Web Software Applications. IEEE Software, vol. 19, no. 2, 2002.

[2] J. C. Knight and E. A. Strunk. Achieving Critical System Survivability through Software Architectures. Architecting Dependable Systems II, Springer, 2004.

[3] IBM. An Architectural Blueprint for Autonomic Computing. Technical Report, 2003.

[4] X. Peng, B. Chen, Y. Yu and W. Zhao. Self-Tuning of Software Systems through Goal-based Feedback Loop Control. The 18th IEEE International Requirements Engineering Conference, 2010.

[5] B. Boehm. Value-Based Software Engineering. ACM SIGSOFT Software Engineering Notes, vol. 28, no. 2, 2003.

[6] A. Dardenne, A. van Lamsweerde and S. Fickas. Goal-Directed Requirements Acquisition. Science of Computer Programming, vol. 20, no. 1-2, 1993.

[7] S. W. Cheng and D. Garlan. Rainbow: Cost-Effective Software Architecture-based Self-Adaptation. Carnegie Mellon University, Pittsburgh, PA, 2008.