Requirements-Driven Runtime Adaptation for Trustworthiness Assurance

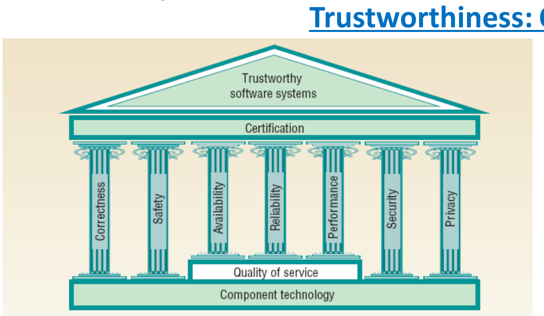
Xin Peng
Associate Professor
School of Computer Science, Fudan University
pengxin@fudan.edu.cn
http://www.se.fudan.edu.cn/pengxin

Introduction

Running in a highly uncertain and greatly complex environment, software systems cannot always provide full set of services with optimal quality, especially when work loads are high or subsystem failures are frequent. To achieve trustworthy system running, developing **self-adaptive systems** with self-management capabilities is promising. In this project, we focus on two aspects of self-management, i.e. self-tuning and self-healing, for runtime trustworthiness assurance. By **self-tuning**, we hope to dynamically tune the priority ranks of functional and non-functional requirements to better adapt to the changing environments and assure survivability. By **self-healing**, our intent is to monitor potential system failures caused by internal faults and external failures and repair the identified problems by compensation, intervention or switching to alternative design.

Self-Adaptation for Trustworthiness Assurance: Background and Idea

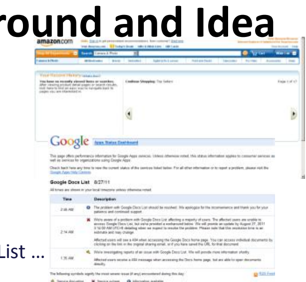
Trustworthiness: Quality Attributes and Survivability



Wilhelm Hasselbring, Ralf Reussner. Toward Trustworthy Software Systems. Computer, April 2006.

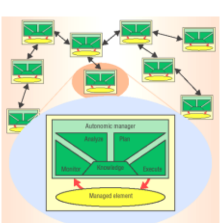
Amazon.com (June 29, 2010)
an outage for more that 3 hours
searching services and shopping carts didn't work
$51,400 loss per minute; shares closed down by 7.8%
http://news.cnet.com/8301-1023_3-20009241-93.html

Google Apps services (just last month)
The affected users are unable to access Google Docs List ...
http://www.google.com/appsstatus#rm=1&di=4&ddo=5&hl=en



Survivability rather than absolute reliability
absolute reliability is often expensive, or even impossible
Survivability [Knight et al. @ 2004]: capability of ensuring crucial services under severe or adverse conditions, with acceptable quality degradation or even sacrifice of some desirable services

Self-Adaptation: Achieve Self-Management by MAPE Control Loop



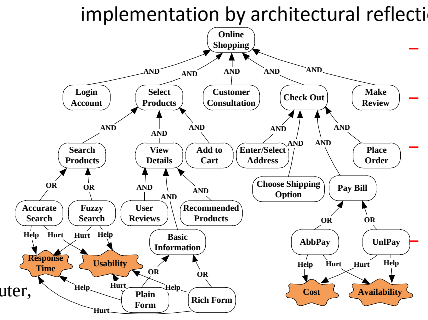
Self-*: *systems shall managing themselves*
- Self-tuning........performance
- Self-configuring...flexibility
- Self-healing.......dependability
- Self-protecting..security/privacy

Requirements-Driven Self-Adaptation
- Requirements models (specifically goal models) as knowledge base for self-adaptation decision making and planning
- Design model and design decision as knowledge base for adaptation implementation by architectural reflection

Self-Adaptation Control Loop

Monitoring
Analyzing
Planning
Execution
\+ Sensing Actuating \+ *Knowledge Base*

Jeffrey O. Kephart, David M. Chess. The vision of autonomic computing. Computer, January 2003.



- Hard goals (rectangle) representing functional requirements
- Soft goals (cloud) representing non-functional requirements
- Alternative solutions are represented as OR decompositions, and their contribution labels imply possible trade off decisions
- Well-formulated reasoning rules for both bottom-up satisfaction propagation and top-down planning

Self-Healing for Potential Failures

- Detect potential failure by runtime verification:
  - pre/post- conditions; temporal specifications; contextual assumption
- Self-repair: resolve potential failures by
  - intervention
  - compensation
  - switching to alternative designs
  - -- switching to other agents providing similar services

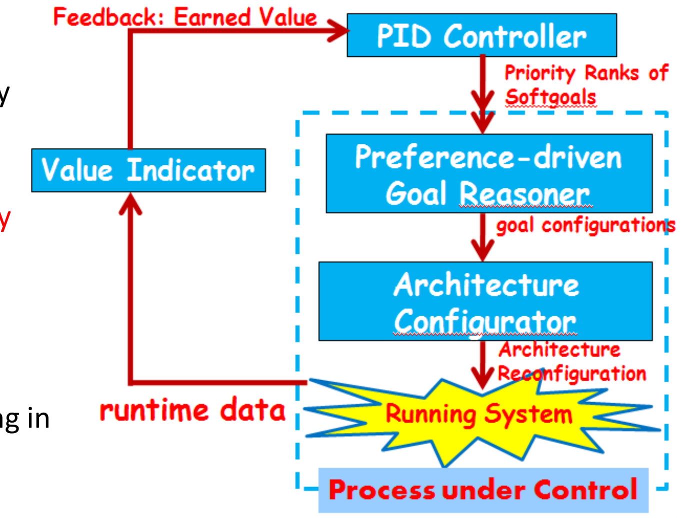
Self-Tuning and Survivability Assurance

Self-Tuning of Software Systems through Dynamic Quality Tradeoff and Value-based Feedback Control Loop

Assumptions
- proper solutions for individual quality attributes
- trustworthiness problems lie in conflicts among different quality attributes

Objective
- achieve optimized overall quality satisfaction by dynamic quality tradeoff at runtime

Solution
- runtime earned value measurement as feedback
- dynamically tuned priority ranks for different quality attributes
- functional requirements reconfigured by requirements reasoning in response to priority tuning of quality attributes
- requirements reconfiguration mapped to runtime architecture



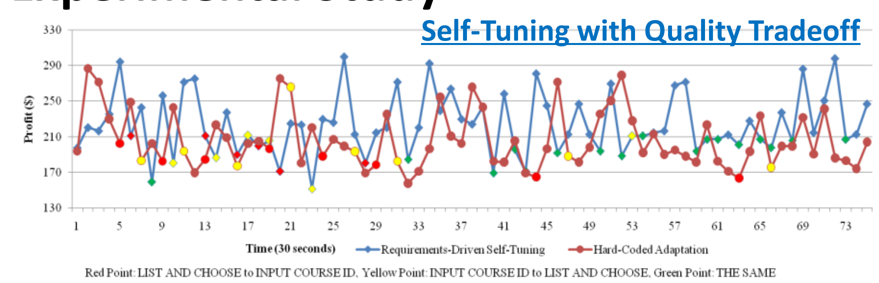
A Step Forward: Self-Tuning for Survivability

Survivability [Knight et al. @ 2004]
- capability of ensuring crucial services under severe or adverse conditions, with acceptable quality degradation or even sacrifice of some desirable services
- survivability rather than absolute reliability: absolute reliability is often expensive, or even impossible

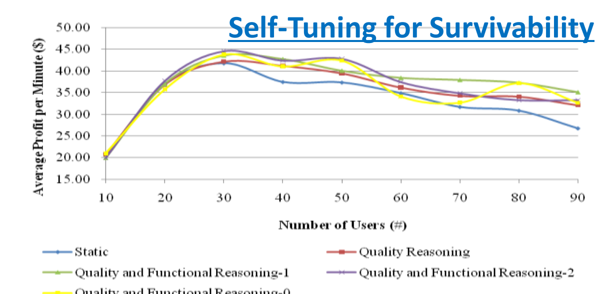
Our Solution: combining functional tradeoff
- functional tradeoff: services (functional requirements) dynamically bound and unbound
- combine quality tradeoff and functional tradeoff
- value-based feedback controller for tradeoff decision

Experimental Study

Self-Tuning with Quality Tradeoff



Self-Tuning for Survivability



Quality and Function Tradeoff-?: quality and function tradeoff with a window of timed delay of ? time units

Future Directions

- Requirements-driven adaptation in more social-technical and distributed applications like mobile, ubiquitous applications, and service oriented systems
- Framework and tools for integration with cloud-based platforms
- Capture and incorporate design decisions as knowledge base for runtime adaptation decisions
- Explore more sophisticated decision mechanisms for runtime adaptations, e.g. control theory, machine learning, AI, …
- Failure diagnosing for more accurate repairing

Conclusion

We use requirements-driven self-adaptation to achieve trustworthiness assurance at runtime. Requirements goal models are employed to represent desired system behaviors, alternative solutions, as well as quality contributions and used as knowledge base for self-adaptation decision making and planning. Intelligent decision mechanisms like feedback control theory are used and runtime earned value is measured as the feedback. Reflective architecture with traceability to requirements is used to map requirements reconfigurations to the structural and behavioral adaptation of runtime architecture.

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

Software Engineering Lab,
Fudan University, Shanghai, China
http://www.se.fudan.edu.cn