Requirements-Driven Runtime Adaptation for Trustworthiness Assurance

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-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

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

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.

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

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

Requirements-Driven Runtime Adaptation

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

Survivability: Google Data Center Outage

Survivability: 99.999% Reliability, $51,400

Survivability: Google Apps services (just last month)