Quality Criteria and an Analysis Framework for Self-Healing Systems

Sangeeta Neti **Hausi A. Müller**
Department of Computer Science
University of Victoria, Canada

ICSE 2007 Workshop on
Software Engineering for Adaptive and Self-Managing Systems (SEAMS)

University of Victoria
British Columbia. Canada

Outline

- Introduction
- Motivation
- Approach
- Quality Criteria for Self-Healing Systems
- Analysis Framework for Self-Healing Systems
- Case Study
- Conclusions

Introduction

- **Autonomic Computing** [Kephart and Chess 2003]

Increased Responsiveness

Adapt to dynamically changing environments

Business Resiliency

Discover, diagnose, and act to prevent disruptions

Operational Efficiency

Tune resources and balance workloads to maximize use of IT resources

Secure Information and Resources

Anticipate, detect, identify, and protect against attacks

Introduction

- **Goal**
  - Develop an analysis and reasoning framework for self-healing systems.
- **Objective**
  - Evaluate architectures of self-healing systems in the context of evolution over long periods of time.
  - Validate and re-assess quality attributes regularly over long periods of time.

Motivation

- **Importance of Architectural Analysis and Evaluation**
  - Architectural analysis is necessary to understand the implications of a design decision.
  - Architectural evaluation is necessary to determine its fitness with respect to certain qualities.

- **Self-Healing System Evolution**
  - Component evolution
  - Change in system usage
  - Change in their designed operating mode

Approach

- **Attribute-Based Architectural Styles (ABASs)**

  [Klein *et al.* 1999]

  - ABASs are an extension of the notion of architectural styles
  - An ABAS consists of
    - A style or an architecture pattern
    - Description of the software components and their relationships
    - Specific quality attribute - Performance, reliability, modifiability, security
    - Analytic framework for reasoning about the quality attribute

Approach

- **Attribute-Based Architectural Styles (ABASs)**

  [Klein *et al.* 1999]

  - Directly related to the evaluation of architecture
  - Collection of ABASs has been created for computing systems
  - ABASs have been used in architecture evaluations and design exercises

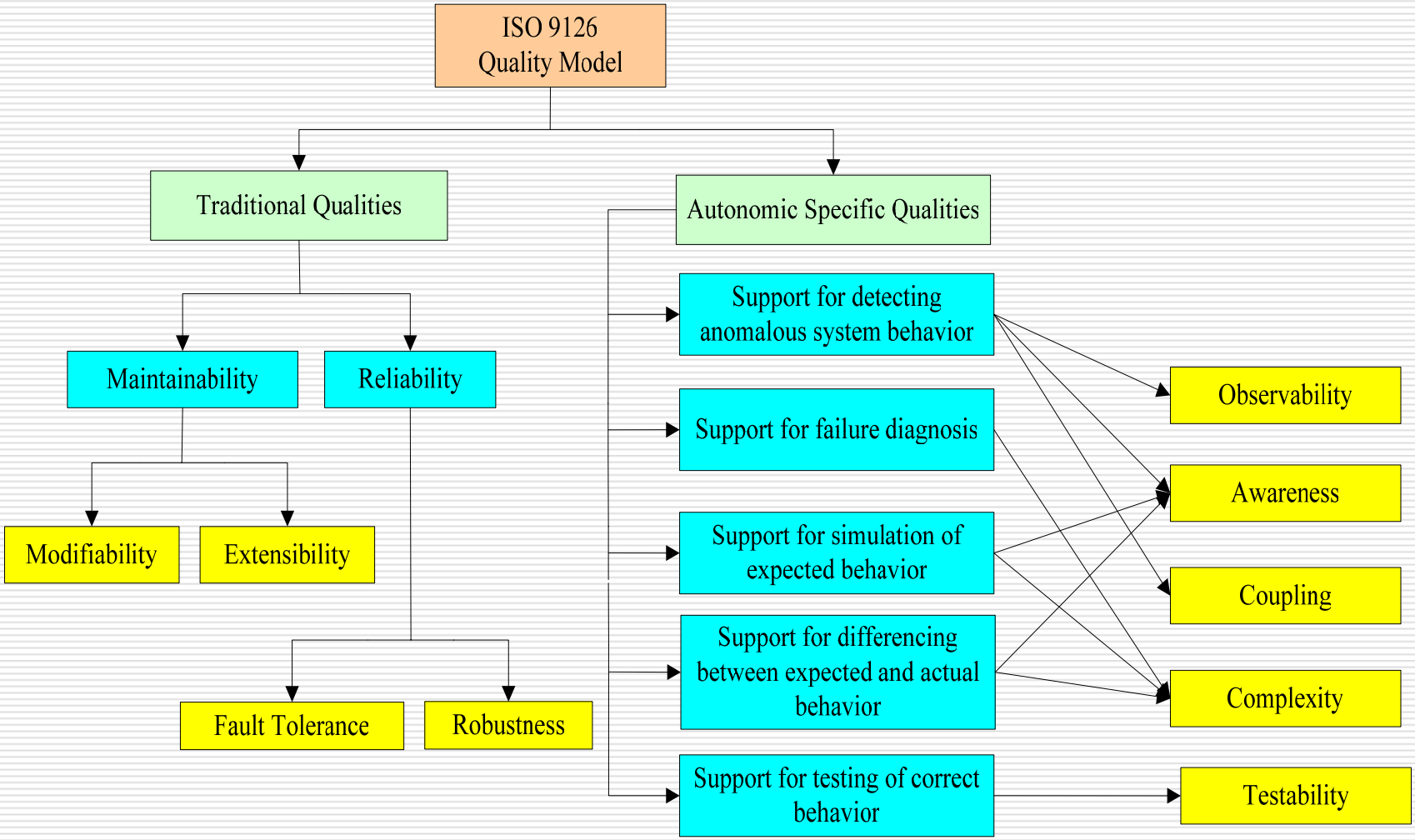
Quality Criteria for Self-Healing Systems

- **Traditional Qualities:** [Salehie and Tahvildari 2005]
  - *Reliability*
  - *Maintainability*

- **New Autonomic-Specific Qualities:**
  - *Support for detecting anomalous system behavior*
  - *Support for failure diagnosis*
  - *Support for simulation of expected behavior*
  - *Support for differencing between expected and actual behavior*
  - *Support for testing of correct behavior*

Quality Model

- **ISO 9126 Quality Model**
  - It proposes a well-defined generic quality model
  - It allows the instantiation of the quality model according to the context
  - It considers internal characteristics and external characteristics
  - It defines six goals: functionality, reliability, usability, efficiency, maintainability, and portability



Quality model for self-healing systems based on ISO 9126

Architectural Styles for Self-Healing Systems

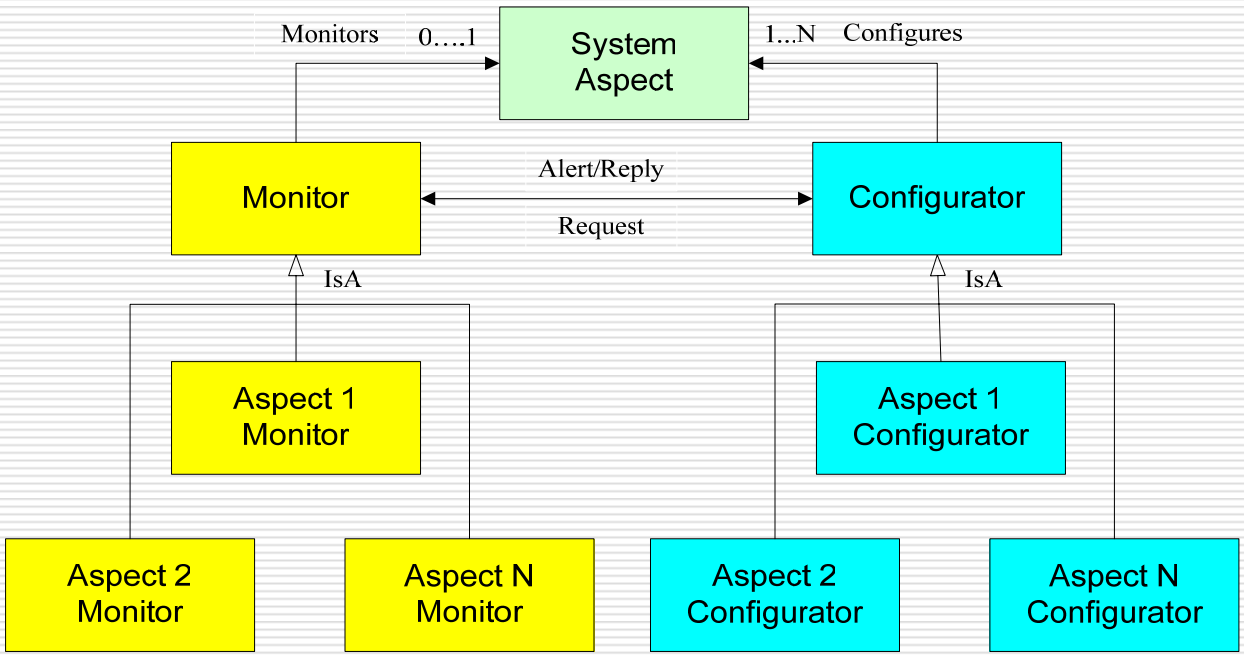
"Architectural styles for Adaptable Self-Healing Dependable Systems"

[Hawthorne and Perry 2005]

- Aspect peer-to-peer architectural style
- Aggregator-escalator-peer architectural style
- Chain-of-configurators architectural style

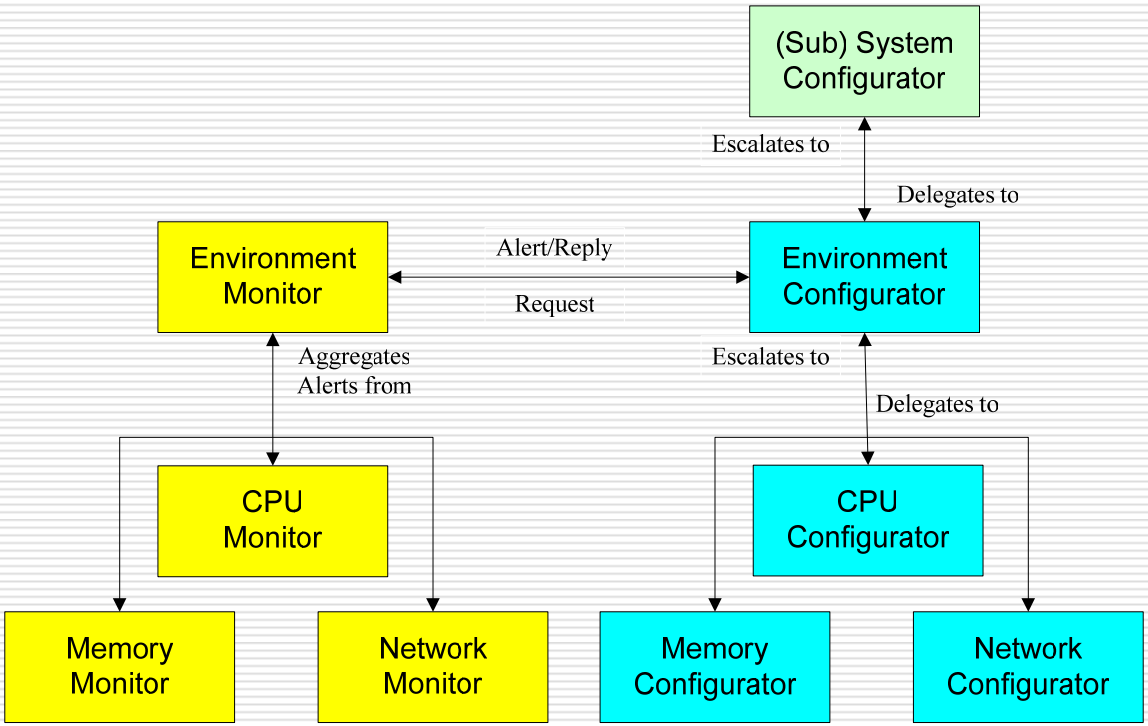
Architectural Styles

Aspect peer-to-peer architectural style



Architectural Styles

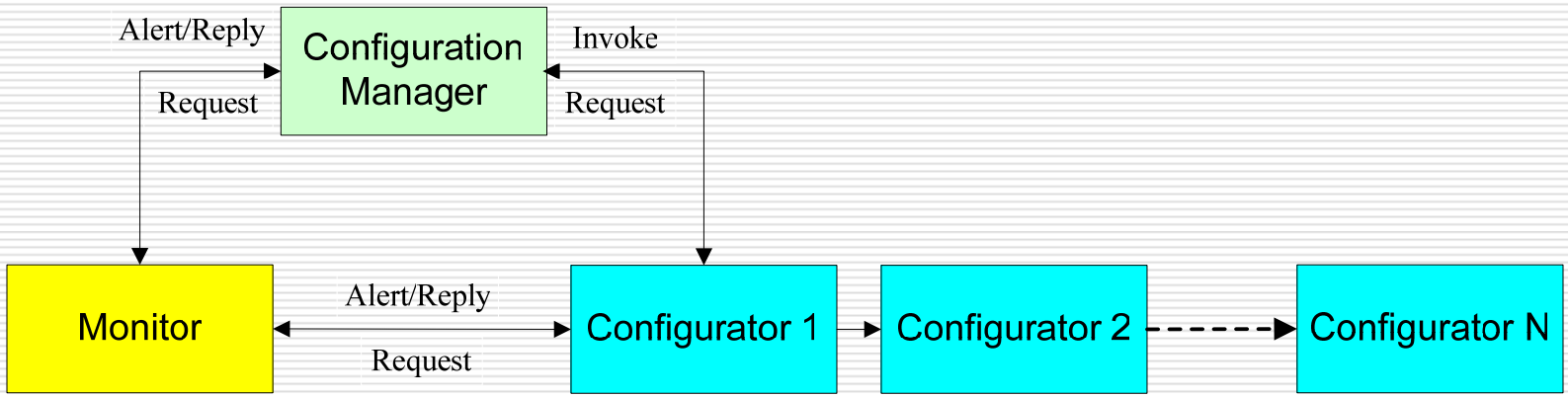
Aggregator-escalator-peer architectural style



Architectural Styles

Chain-of-configurators architectural style

- Chain of responsibility design pattern [Gamma *et al.* 1995]

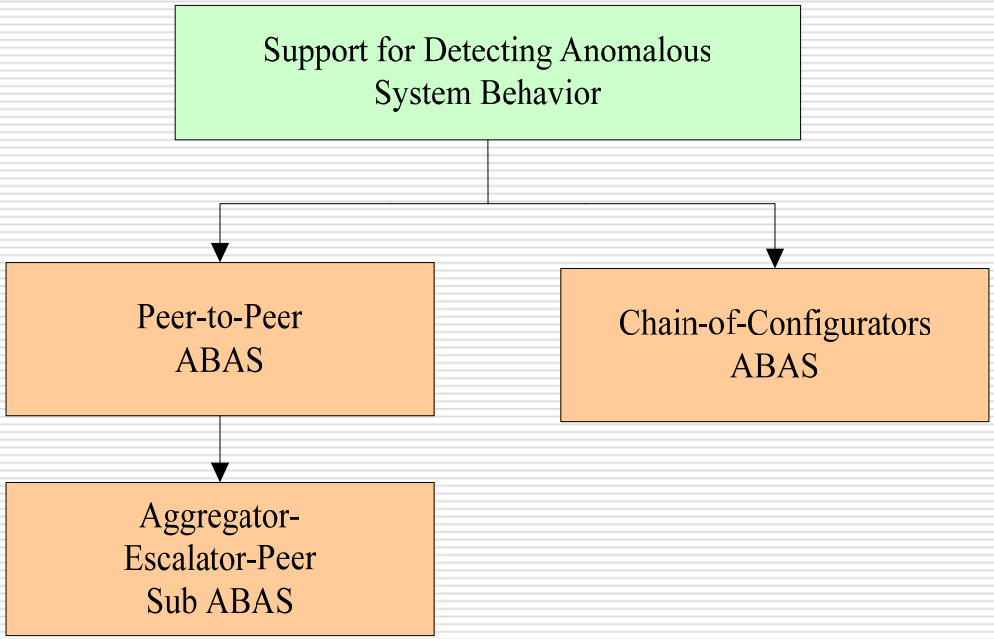


Analysis Framework for Self-Healing Systems

- **Analysis using ABAS**
  - **Structure of an ABAS** [Klein *et al.* 1999]
    1. **Problem description**
    2. **Stimulus/Response attribute measures**
    3. **Architectural style**
    4. **Analysis**

Analysis Framework-Example

Autonomic-Specific Quality ABAS



Characterization of the Support for Detecting Anomalous System Behaviour ABAS

Analysis Framework- Example

- **Peer-to-peer support for detecting anomalous system behavior ABAS**

  - **Stimulus:** A fault in the system.
  - **Response:** Is the system detecting the anomaly caused due to this fault?

  - **Measurable Quantities:**
    - *Detection rate and time*
    - *Coupling*
    - *Awareness*
    - *Observability*
    - *Fault model*

Analysis Framework- Example

- **Analysis:**
  - ***Coupling***: No dependencies between peers.
  - ***Awareness***: Direct binding of components with the monitors. Monitors do not interact with each other.
  - ***Observability***: Architecture is easily modifiable. Easier to attach to environment facilities.
  - ***Fault model***: Suitable for small systems.

Analysis Framework-Example

- **Aggregator-escalator-peer support for detecting anomalous system behavior Sub ABAS**
  - **Analysis:**
    - ***Coupling***: Dependencies between peers.
    - ***Awareness***: Direct binding of components with the monitors. Monitors interact with each other.
    - ***Observability***: Architecture not easily modifiable. It may not be easier to attach to environment facilities.
    - ***Fault model***: Suitable for large scale systems.

Analysis Framework- Example

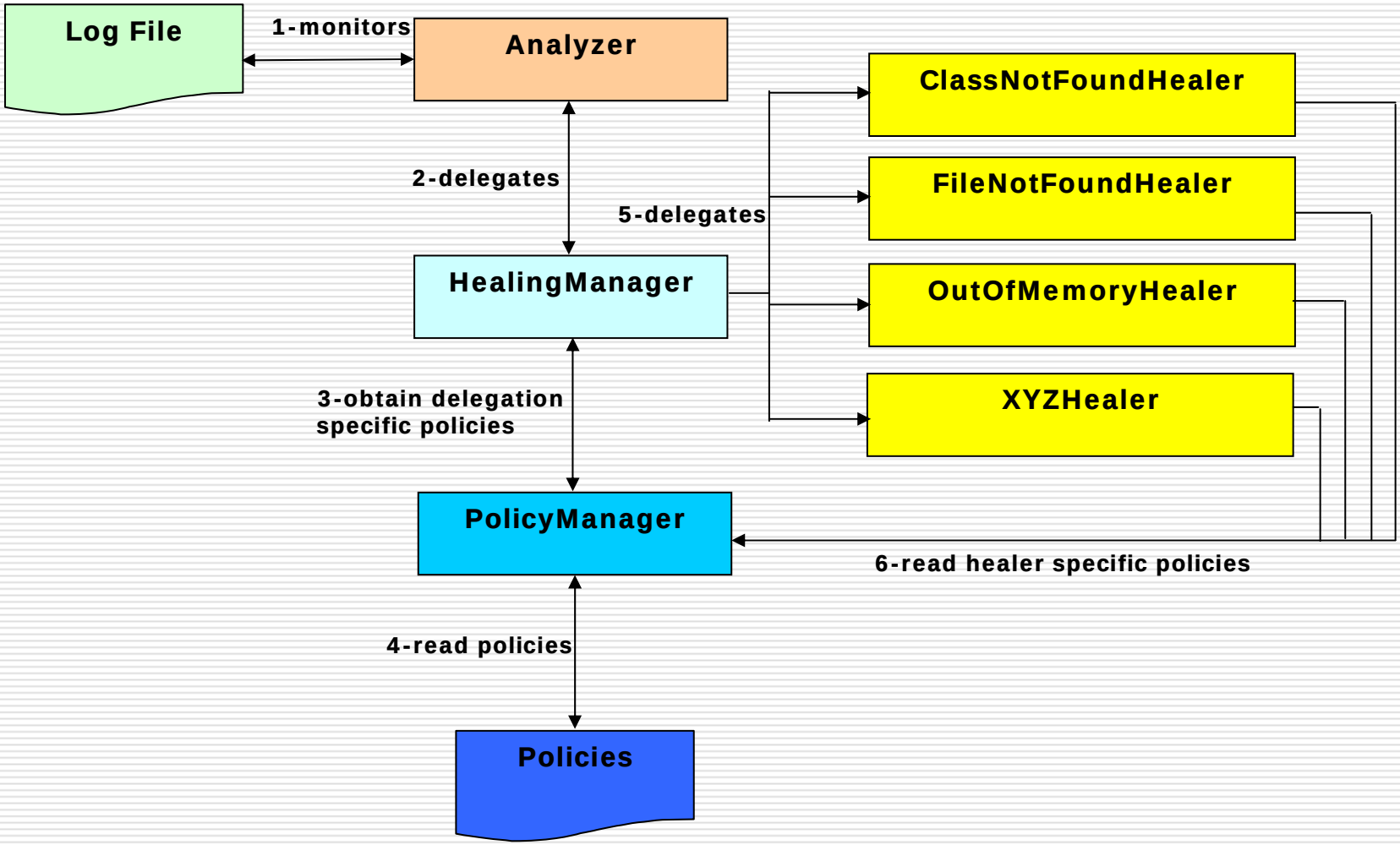
- **Chain-of-configurators support for detecting anomalous system behavior ABAS**
  - **Analysis:**
    - ***Coupling***: Architecture enhances loose coupling.
    - ***Awareness***: No separate monitor and configurator for each component of the system.
    - ***Observability***: Architecture enhances run-time modifications. Easier to attach to environment facilities.
    - ***Fault model***: Suitable for different kinds of faults. An optimum strategy can be chosen for a given problem.

Case Study

- **Model for Self-Managing Java Server**

  [Kumar and Rao 2003]

  - It is a working model of a non-stopping Java-based server
  - The server has self-configuration and self-healing capabilities



Message Flow in the Java server model

Case Study

- **Analysis**

Design Decision: The architecture enhances loose coupling.

Architectural Style:

- Chain-of-configurators architectural style

Quality attributes:

- *Modifiability*
- *Support for detecting anomalous system behavior*
- *Support for failure diagnosis*

Case Study

- **Implications:**
  - Modifiability comes at the cost of performance.
  - Easy to reconfigure.
  - ***Coupling:*** The rate and time of anomaly detection is potentially better.
  - ***Awareness***: Lack of self-awareness.
  - ***Observability:*** Easier to attach to environment facilities.
  - ***Fault model***: It can handle a variety of exceptions.
  - ***Diagnosis rate:*** Diagnosis becomes easier.

Conclusions

- **Summary**
  - Developed an analysis and reasoning framework for self-healing systems.
  - The proposed framework can facilitate the design and maintenance of self-healing systems.

- **Future Work**
  - The proposed reasoning and analysis framework can be extended to other self-managing applications.
  - The relationship between architecture and quality attributes of self-managed systems can be recorded using a reverse engineering handbook.

Conclusions

- **Contributions**
  - Defined quality criteria for self-healing systems
  - Customized the ISO 9126 quality model
  - Developed framework with respect to traditional as well as autonomic-specific quality attributes.

- **Publication**

  S. Neti and H.A. Müller, “Quality Criteria and an Analysis Framework for Self-Healing Systems,” *Proceedings ACM/IEEE ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2007)*, Minneapolis, Minnesota, May 2007 (In Press).

Thank You!