Elements of the Self-Healing System Problem Space

Phil Koopman
Carnegie Mellon University

WADS, May 2003
Overview

“Self-Healing” – it’s getting attention, but what does it mean?

- This talk is based on observations from the most recent Workshop on Self-Healing Systems (WOSS’02)

Description of some general problem elements of Self Healing research

- Fault models – what is an “injury”?
- System responses – what is “healing”? 
- System incompleteness – what’s unknown?
- Design context – what injuries are beyond healing?

Two challenges:

1. Fault Tolerant Computing: broaden perspectives with SH ideas
2. Self Healing: don’t waste time reinventing existing FT ideas
Fault Model – “injury”

First question in fault tolerant computing is:

“What is the fault model?”

Reasons for a fault model

- Need to know expected faults to measure fault tolerance coverage
- Not all faults are equal in time, space, severity

Some challenges:

- Is Injury == Fault
- Is a software defect an injury?
Self-Healing Fault Model Issues

- **Fault duration:**
  - Permanent / intermittent / transient

- **Fault manifestation:**
  - Fail silent / Byzantine / correlated faults
  - Impaired: run-time, reserve capacity, brittleness, resource consumption

- **Fault source:**
  - Wear-out / design defects / reqts. defects / environment change / malicious

- **Granularity:**
  - One designer’s “system” is the next level designer’s “component”
  - Transistor failure / … node failure … / system failure

- **Fault profile expectations:**
  - No faults / historically known faults / foreseen faults / unforeseen faults
  - Random+independent / random+correlated / expected / predicted
System Response – “healing”

- After an injury, what happens?

- Fault tolerant system responses include:
  - Diagnosis / identification
  - Isolation / containment
  - System reconfiguration
  - System reinitialization

- Does “healing” mean something additional?
  - Or is it a difference at a different level?
Self Healing System Responses

- **Fault Detection:**
  - Self-test / pairwise checking / peer checking / supervisor checking
  - Self-injected faults to ensure detection is working?

- **Degradation during & after healing:**
  - Fail-operational / degraded performance / fail-fast+ fail-safe

- **Response:**
  - Fault masking / failover / reconfiguration
  - Optimize for: safety / reliability / availability / …
  - Preventative (periodic reboot) / Proactive (diagnosis-based) / Reactive

- **Recovery of state:**
  - Hot swap / restore quiescent state / warm boot / cold boot
  - Rollback / recovery block / control gain changes / rollforward / run-while-reconfiguring
  - What about recovering component state?

- **Time constants:**
  - Most faults are transient
  - Important that system response time constant be faster than injury arrival rate

- **System Assurance:**
  - After injury / during healing / after healing
System Completeness – *What do we know and when?*

- **System self-knowledge**
  - How much self-knowledge is required for healing?
  - How should healing knowledge be abstracted?
  - How do we deal with not knowing how much the system doesn’t know?

- **Designer knowledge**
  - Not all systems are complete when design is “done”
  - Even if complete, we won’t know everything about all components
  - How do we deal with not knowing how much we don’t know?
Self Healing System Completeness

- Architectural Completeness:
  - Proprietary & known / open & regulated / extensible

- Designer Knowledge:
  - Component knowledge (especially COTS components)
  - Faulty behavior characterizations
  - How do you heal after suffering a component behavior that is “unspecified”?

- System Self-Knowledge:
  - How complete is system’s self-model? (idea of reflection)
  - Is healing an intentional or emergent behavior?

- System Evolution
  - Configuration changes & usage changes
  - Are outages random / predictable / schedulable?
Design Context — What are the scope limits?

- The real world is a messy place – what assumptions are made?
  - Homogeneous system?
  - “Perfect” components (e.g., perfect healing management software?)
  - …

- What is the size of the system?
  - A single software module?
  - A complex software system?
  - A person plus a computer system?
  - The North American power grid?
  - The Internet?
  - Does teaching users to press CTL-ALT-DEL achieve “self-healing” of the user+computer “system”?
Self Healing Design Context

- **Abstraction Level:**
  - Implementation / design / architecture / …

- **Component Homogeneity:**
  - Can any software component run in any node?
  - Perfect configuration homogeneity / plug-compatible / heterogeneous

- **Predetermination of system behavior:**
  - Specific design / rule-based system / service discovery / emergent behavior

- **User Involvement in healing:**
  - User direction / user-provided hints / user ability to tune / invisible to user

- **System Linearity:**
  - Linear+composable / monotonic / mildly discontinuous / arbitrary
  - Single operating mode / mode changes

- **System scope:**
  - Component / computer system / computer+person / enterprise / society
Conclusions

• “Self-Healing” potentially encompasses a lot of ground
  • Smaller than expected intersection of research assumptions at WOSS02
  • Consensus will take a while

• Some of this has been done before!
  • Fault models – well known in FT, don’t reinvent without good reason
  • System responses – how different are they from FT?
  • System incompleteness – FT usually assumes relative completeness
  • Design context – plenty of room for novelty in both FT & SH

  • But there is plenty of room for more good research

• A final thought:
  1. Fault Tolerant Computing: broaden perspectives with SH ideas
  2. Self Healing: don’t waste time reinventing existing FT ideas
     even better: articulate the novelty of approaches