ACM SIGSOFT* Workshop on Self-Healing Systems (WOSS'02)

November 18-19, 2002
Charleston, SC

* With partial support from the NASA High Dependability Computing Project

Workshop Logistics

Sessions organized by affinity groups

- 3-4 papers per 1.5 hour session

Session chairs, as noted on program

- See your chair before your session

Emphasis on discussion

- 10-minute research overviews from each paper

Half-hour breaks between sessions

Joint lunch on Monday

Open (uncommitted) sessions at end of each day (for extra discussion, topics that don’t have another home, etc.)

Software Engineering Today

Common assumptions

- Known and stable system requirements
- Known and stable operating environment
- Control over the development of assembled parts
- Development time and run time are completely separate
- Systems can be taken “down” for “maintenance”

Consequences

- Focus on improving development time processes, techniques, notations
- Provide high assurance through testing, rigorous specification, modeling, verification, etc.
- Expect end users to do manual installation and upgrades

Isn’t This Good Enough?

Increasingly, systems

- are composed of parts built by many organizations
- must run continuously
- operate in environments where resources change frequently
- are used by mobile users

For such systems, traditional methods break down

- Exhaustive verification and testing not possible
- Manual reconfiguration does not scale
- Off-line repair and enhancement is not an option

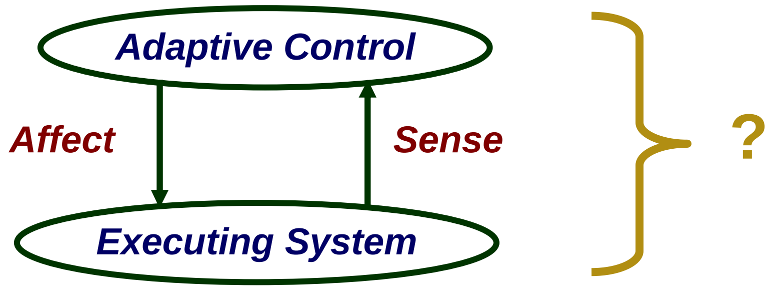
What Has to Change?

Goal: systems automatically and optimally adapt to handle

- **changes in user needs**
- **variable resources**
- **faults**
- **mobility**

But how?

Answer: Move from open-loop to closed-loop systems



Many Approaches

Programming language support

Algorithms (e.g., self-stabilizing, machine learning)

Architecture-based adaptation

Operating systems support

Domain-specific techniques (e.g., distributed databases, pub-sub architectures, …)

Adaptable middleware

Support for user mobility

Fault tolerant system design (e.g., graceful degradation)

Biologically-inspired models

Inferring correct system behavior through observation

Why Have a Workshop?

Understand the relationships between these different approaches

Identify the *software engineering* challenges and opportunities

Create a common vocabulary (or possibly a reference model)

Affinity Groups (in order of appearance)

Architecture-based adaptation

Systems: operating systems, distributed systems, databases

Middleware & mobility

Programming languages

User-centric approaches: requirements specification, inference

New paradigms: neural nets, biologically-inspired computing, homeostatic systems