Software Architecture

Lecture 11
Adaptation and Self-Adaptation

João Pedro Sousa
George Mason University

previously
analysis as a tool to manage tradeoffs

- early elicitation of the system's key characteristics
- multiple analysis techniques complement each other
- *how much analysis?*
  key aspect of an architect's job
  - too much will expend resources unnecessarily
  - too few risk allowing bad decisions into the final system
  - wrong kinds of analyses will have both drawbacks
- the benefits typically far outweigh the costs

today
adaptation

- characterize change
  - reasons for change
  - what changes
  - when/how it changes
- architecture-centric adaptation
  - feedback control loop
- self-adaptation

change
comes with the software *territory*

- reasons for change
  - stakeholders want to
    - customize a new version of entire system
    - add new features to an existing system
    - improve existing features/fix problems
    - remove/restrict access to features
    - improve some QAs
  - application environment changes
    - competition has cool new features
    - platforms/OS/devices evolve
    - resources/load fluctuate during operation

what changes

- code
  - component/connector internal logic
  - component/connector APIs
    - i.e. traditional software maintenance
- parameters of execution/configuration
  - modes of operation in embedded systems
    - e.g. elevators, HVAC, robots, cars…
- C&C run-time structure
  - change connectors while maintaining style
    - e.g. local -> distributed system
  - change features
    - e.g. change service coordination to invoke new kinds of services
  - same features, tune QAs
    - e.g. remapping of service providers, # of replicas of a web server…

when/how it changes

- code
  - offline, by human hands, except self-modifying code
- parameters of execution/configuration
  - decision made by
    - humans
    - automatically based on sensor readings
  - changes effected automatically
- C&C run-time structure
  - change features/architectural constraints
    - offline, by human hands
  - same features, tune QAs
    - offline, by human hands
    - automatically based on human-defined reqs/goals

autonomic computing
aka self-* software

architecture models
help with changes

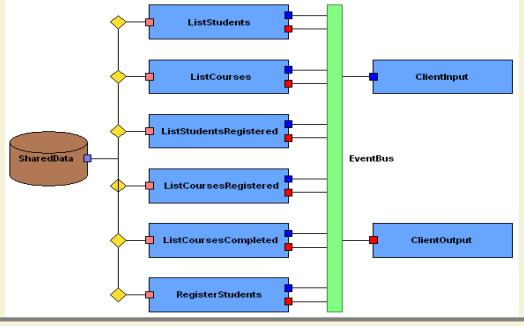
- code
  - offline, by human hands
- parameters of execution/configuration
  - decision made by
    - humans
    - automatically
  - changes effected automatically
- C&C run-time structure
  - change features/architectural constraints
    - offline, by human hands
  - same features, tune QAs
    - offline, by human hands
    - automatically

analyze
scope of impact
effects on QAs…

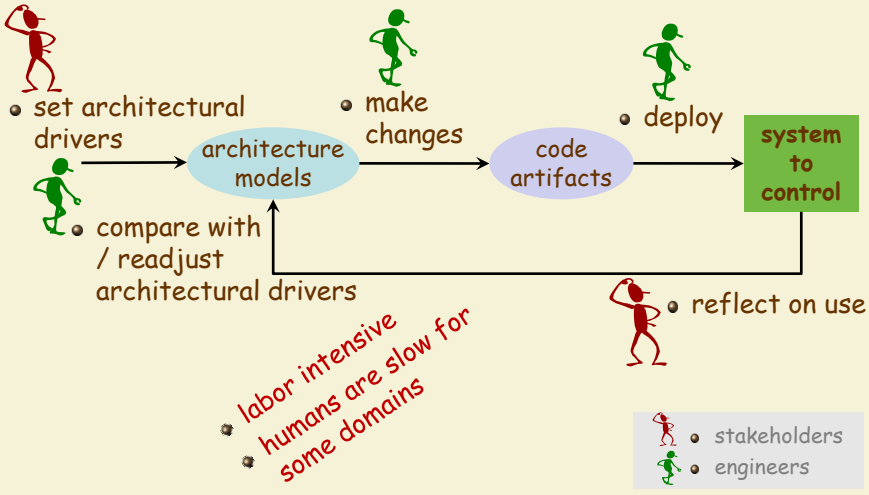
example change architectural constraints
make systems distributed

- data streaming system based on Java pipes
  e.g. lab1

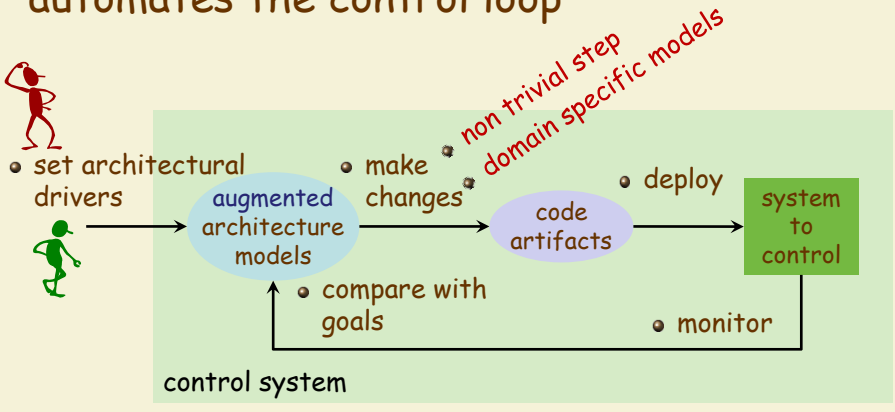
- pub-sub system based on Java Observer-Observable



adaptation as a feedback control loop

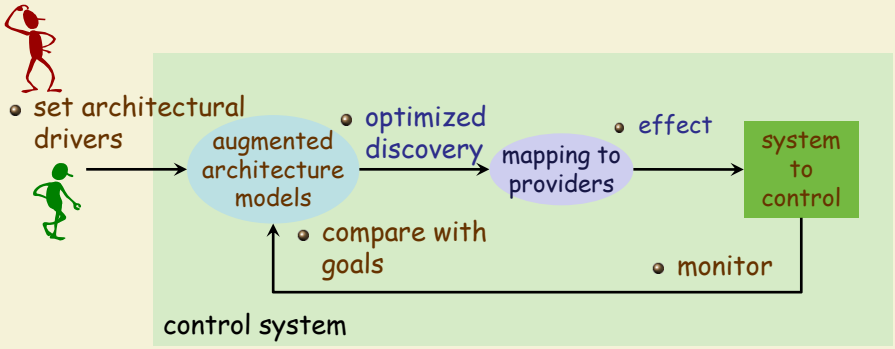


self-adaptation automates the control loop



- systems with understood adaptation patterns
  - mission critical
  - 24/7
  - embedded

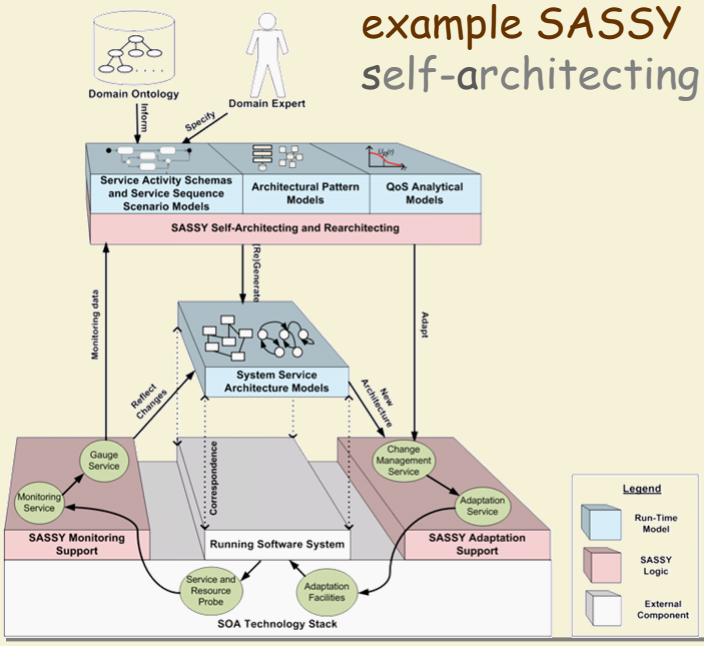
self-adaptation made easier by discovery in SOA



- systems with understood adaptation patterns
  - mission critical
  - 24/7
  - embedded

stakeholders
engineers

example SASSY self-architecting systems



in summary
change is part of business

- traditional: manual changes
- emerging: automated changes
  aka self-adaptation
  - feedback control loop
    - automated discovery
    - automated QA analyses