PRESENTATION ON SELF-MANAGED SYSTEMS : AN ARCHITECTURAL CHALLENGE

BY: JEFF KRAMER AND JEFF MAGEE
DEPARTMENT OF COMPUTING IMPERIAL COLLEGE LONDON
SW7 2AZ, UK
{J.KRAMER,J.MAGEE}@IC.AC.UK

Synopsis by: Parnian Najafi
What is self-managed software architecture?

- Components automatically configure their intercommunication based on an overall architectural specification in order to achieve the goals of the system, with minimum explicit management.
Self Managed Systems

- Self-configuration
- Self-adaptation
- Self-healing
- Self-monitoring
- Self-tuning

Self-* or autonomic systems
Self-configuration

- Components should configure themselves to satisfy specification or report that they cannot
Self-healing and self-adaptation

- In the case of changes in the requirement specification, operational environment, resource availability or faults in the environment or system:
  - Reconfigure
  - Degrade gracefully
  - Report an exception
Why an architecture-based approach?

- Generality
- Level of abstraction
- Potential for scalability
- Builds on existing work
- Potential for an integrated approach
Others use architectural approach too

- Oreizy: uses architectural approach-adaptation and evolution management
- Garlan and Schmerl: use architectural models for self-healing
- Dashofy, van der Hoak and Taylor use architecture evolution manager for run-time adaptation and self-healing in ArchStudio
- Gomaa and Hussein: use of dynamic software reconfiguration and reconfiguration pattern for software product families

...
Architectural Model for Self-management

- Robotics
- Sense-plan-act (SPA)
- Garlan’s self-healing system:
  - Monitoring
  - Analysis/resolution
  - Adaptation
- Gat:
  - Control: Reactive feedback control
  - Sequencing: Reactive plan execution
  - Deliberation: Planning
Control Layer

- Consists of:
  - Sensors
  - Actuators
  - Control loops
Control Layer Responsibilities

- Self-tuning
- Event and status reporting to higher levels
- Operations to support modification
- Component addition, deletion and interconnection
- When the current configuration of components is not designed to deal with a situation, the layer detects this failure and reports it to higher layers.
Sequencing Layer

- Reacts to changes in state reported from lower levels
- Execute plans with new control behaviors and new operating parameters for existing control layer behavior
- Execute an action or sequence of actions to handle the new situation
Sequencing Layer Capabilities

- Introduce new components
- Recreate failed components
- Change component interconnections
- Change component operating parameter
Sequence Layer Characteristic

- Essential characteristic of change management layer is that it consists of a set of pre-specified (pre-computed) plans which are activated in response to state change.
- If a situation is reported for which a plan does not exist, this layer must invoke the services of a higher planning layer.
Goal Management (Deliberation)

- **Time consuming computation**
  - Planning based on the current state to achieve the specification of high level goal
    - i.e. By current position of the robot and map of its environment -> producing a route plan for execution by sequencing layer
    - Changes like obstacles that are not in the map cause re-planning

- **Produces change management plans according to requests from the layer below and introduction of new goals**
Three layer model of self-managed systems

- Immediate feedback actions at the lowest level and the longest actions are at the top level
Component Control Layer

- A component implements the set of services that it provides (may use other services to implement them)
- Mode: abstracted view of internal state of a component
Operations on Components

- create C: T
  - create component instance C from type T.
- delete C
  - delete component instance C.
- bind C₁.r – C₂.p
  - connect required port r of component C₁ to provided port p of component C₂.
- unbind C₁.r
  - disconnect required port r of component C₁
- set C₁.m to val
  - set mode m of component C₁ to val.
Research Challenge in Component Control Layer

- Preserving safe application operation during change.
  - i.e. change in a mechatronic system controlling a vehicle.
Change Management Layer

- Responsible for executing changes in response either to changes in state reported from the lower layer or in response to goal changes.
- This layer is a precompiled set of plans and tactics that respond to a predicted class of state change.
  - i.e. in fault tolerant system, failure of a component may cause a duplicate server to immediately switch from standby to active. Change management should make another standby server.
Research Challenge

- Distribution and decentralization defines the difference between self-management of complex software systems and existing work on robotic systems.
- Distribution raise issues like:
  - Latency
  - Concurrency
  - Partial failures
- Coping with distribution and arbitrary failures lead to the need for some level of local autonomy while preserving global consistency.
Distribution and Decentralization are troublemaking!

- Due to distribution obtaining consistent view of the system state to make change decisions is hard.
- Decentralization of control makes robust execution in cases with partial failure, difficult.
To solve these

- A decentralized change management architecture makes state changes to be serialized to make sure configuration terminate in a valid state.
Change management functionality is included with each component.

Each component maintained a view of an overall system and executed local changes in response to state changes in the view.

Problems:
- The view of the system has to be complete
- Requires a total order broadcast bus to keep views consistent
The architecture was a fully decentralized architecture that reliably executed change in the presence of arbitrary failure. It was not a scalable architecture.

- i.e. systems that can accommodate partial inconsistent views and relax the need for totally ordered broadcast communication.

- Finding change management algorithm that can tolerate inconsistency and will eventually terminate in a system that satisfy constraints.

- The system should not violate safety constraints while it is converging on a stable state.

- Self stabilizing algorithms have specific configuration and application.
Since we wanted to preserve the global structural constraints, a consistent view of system architecture was necessary. A more behavioral view of the system constraints will provide opportunity for relaxing the consistency requirement. If we are not interested in architecture, components can bind to any service that satisfies the local requirement. Failure of the remote service can trigger a search for replacement service.
Goal Management Layer

- Precise specification of both application goals and system goals
- Refinement from high-level goals to specified goals (processable by machines) with human assistance
Challenge

- Goal specification that it is both comprehensive by human users and machine readable.
- Producing a change plan based on system goals and current state of the system
- May be intractable problem
- If tractable, response time may be an issue
Solutions

- Design a set of plans offline
- Try them
- Will the change plans satisfy any possible system states?
  - used in active-standby server pairs
- Challenge: provide online planner, when change management layer figure out non of the current plans apply to observed system state
- In decomposition of goals, operational plan from the goals, constraining the problem domain helps.
Conclusion

- Self management at the architectural level.
- In self-managed SW architecture, components automatically configure their interaction in a way that is compatible with an overall architecture specification and achieves the goals of the system.
Conclusion

- **Component Layer:**
  - Provide change management that:
    - reconfigures the software components
    - ensures application consistency
    - avoids undesirable transient behavior.

- **Change Management Layer,**
  - Decentralized configuration management
    - Can tolerate inconsistent views of the system state
    - Converge to a satisfactory stable state.

- **Goal Planning Layer:**
  - On-line (perhaps constraint based) planning
Overall Challenge

- A comprehensive solution based on integrated solutions to the challenges supported by an appropriate infrastructure.