Welcome to SENG 480B / CSC 485B / CSC 586B Self-Adaptive and Self-Managing Systems

Dr. Hausi A. Müller
Professor
Department of Computer Science
University of Victoria

http://courses.seng.uvic.ca/courses/2013/summer/seng/480b
http://courses.seng.uvic.ca/courses/2013/summer/csc/485b
http://courses.seng.uvic.ca/courses/2013/summer/csc/586b

1

Announcements

- Fri, June 28
  - Midterm in class
  - Prof. Venkatesh Srinivasan will administer the midterm
  - I will grade it ☺
- Midterm format
  - Closed books, closed notes, no phones or gadgets
  - Mostly essay type questions
- Midterm topics
  - Self-adaptive systems
  - Autonomic systems
    - MAPE-K loop
    - ACRA hierarchy
    - Symptoms
    - Policies
  - Feedback systems
    - Positive/negative/hybrid feedback
    - PID controllers
  - ULS systems
    - ULS characteristics
    - Wicked problems

2

Policy Types for Autonomic Computing

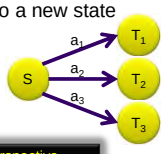
- Action policies
  - **If-then action rules** specify exactly what to do under the current condition.
  - Rational behaviour is compiled in by the designer
  - Basis for reflex agents
- Goal policies
  - Requires self-model, planning, conceptual knowledge representation
- Utility function policies
  - It chooses the actions to maximize its utility function
  - Finer distinction between desriability of different states than goals
  - Numerical characterization of state
  - Needs methods to carry out actions to optimize utility

Real autonomic systems embody a combination of policy types.

3

Action Policies

- A state *S* is a vector of attributes
- *S* can directly be measured by a sensor, or
- *S* can be inferred or synthesized from lower-level measurements
- Policy will directly or indirectly cause an action *a*
- Deterministic or probabilistic transition into a new state from *S* to a new state *T*

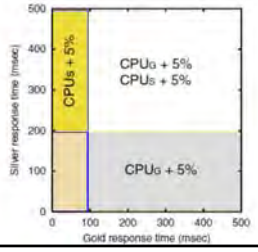


Kephart and Walsh: An Artificial Intelligence Perspective on Autonomic Computing Policies, POLICY 2004.

4

Action Policy Example

- RT: Response Time
  if (RT_{Gold} > 100 ms) increase CPU_{Gold} by 5%
  if (RT_{Silver} > 200 ms) increase CPU_{Silver} by 5%



5

Action Policy Examples

- For each machine, if idle session is greater than 20 minutes then terminate the session
- BitTorrent user processes initiated from IP address 141.223.2.15 should have lowest priority

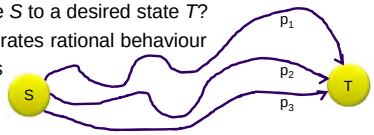
if (srcIPaddress == 141.223.2.15) && process-type == "bittorrent")
then priority is low

- **Event**
  - Total number of user logins is greater than 5 **and**
  - CPU load is greater than 90 **and**
  - Total number of processes running is greater than 35
- **Action**
  - Block any new user logins

6

Goal Policies

- Instead of specifying what to do in a current state, specify a single desired state or a set of desired states
- Any member of this target set is equally acceptable
- Cannot express fine distinctions in preference
- How to compute a set of actions that gets the system from current state *S* to a desired state *T*?
- The system generates rational behaviour
- Potential conflicts
- How to resolve conflicts?

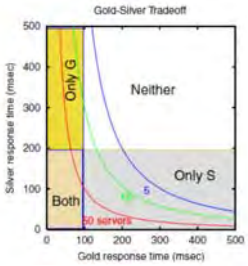


Kephart and Walsh: An Artificial Intelligence Perspective on Autonomic Computing Policies, POLICY 2004.

7

Goal Policy Example

- RT: Response Time
  Gold: RT_{Gold} <= 100 ms
  Silver: RT_{Silver} <= 200 ms



Kephart and Walsh: An Artificial Intelligence Perspective on Autonomic Computing Policies, POLICY 2004.

8

Utility-Function Policies

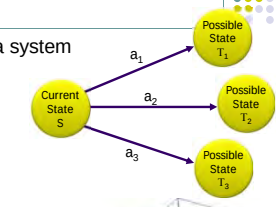
- An objective function to express the value of each possible state
- Generalizes goal policies
- Instead of desirable/undesirable we have a real-valued scalar desirability for each state
- More fine-grained and flexible specifications
- Goal functions often exhibit conflict; use utility function to resolve conflict
- Unambiguous, rational decision making

Kephart and Walsh: An Artificial Intelligence Perspective on Autonomic Computing Policies, POLICY 2004.

9

Utility Functions

- Map any possible state of a system to a scalar value
- Obtained from
  - Service Level Agreement
  - Preference elicitation
  - Simple templates
- Useful representation for high-level objectives
  - Value can be transformed to guide system behavior

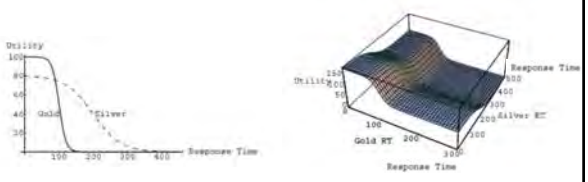


Kephart and Walsh: An Artificial Intelligence Perspective on Autonomic Computing Policies, POLICY 2004.

10

Utility-Function Policy Example

- U: Utility function
  $U(RT_{Gold}, RT_{Silver}) = U_{Gold}(RT_{Gold}) + U_{Silver}(RT_{Silver})$



Kephart and Walsh: An Artificial Intelligence Perspective on Autonomic Computing Policies, POLICY 2004.

11

Policy Types for Autonomic Computing

- Real autonomic systems embody a combination of policy types
- In ACRA
  - Lower levels typically use action policies
  - For higher levels, goal or utility-function policies are more appropriate
- Unified framework is needed to support multiple policy types within a single autonomic component

Kephart and Walsh: An Artificial Intelligence Perspective on Autonomic Computing Policies, POLICY 2004.

12

Policy Model and Terminology

- **Policy**
  - A collection of policy alternatives.
- **Policy Alternative**
  - A collection of policy assertions.
- **Policy Assertion**
  - Represents an individual requirement, capability or other property of a behavior.
- **Policy Assertion Type**
  - Represents a class of policy assertions and implies a schema for the assertion and assertion-specific semantics.
- **Policy Assertion Parameter**
  - Qualifies the behavior indicated by a policy assertion.
- **Policy Vocabulary**
  - The set of all policy assertion types used in the policy.
- **Policy Expression**
  - An XML Infoset representation of a policy.
- **Policy Subject**
  - An entity (e.g., an endpoint, message, resource, interaction) with which a policy can be associated.
- **Policy Scope**
  - A collection of policy subjects to which a policy may apply.
- **Policy Attachment**
  - A mechanism for associating policy with one or more policy scopes.

13

IBM's Autonomic Computing Mission

Mission: Address IT complexity in heterogeneous environments through self-managing autonomic capabilities

Common overarching architecture	Autonomic standards	Common technology for end-to-end management

Autonomic capabilities in IBM and industry products

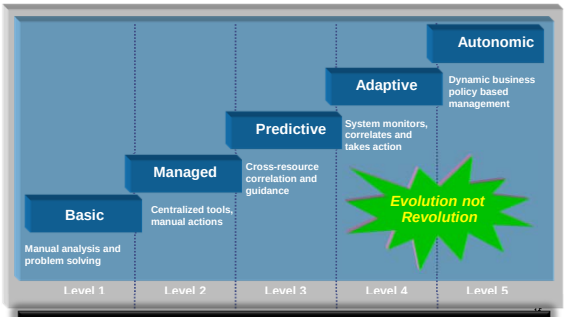
Enabling intelligent systems that:
- Adapt to unpredictable conditions
- Continuously tune themselves
- Prevent and recover from failures
- Provide a safe environment

Empowering businesses to:
- Reduce downtime
- Increase productivity and savings
- Improve resiliency and agility
- Accelerate time to value

David Bartlett, VP: IBM Autonomic Computing, ICAC 2006

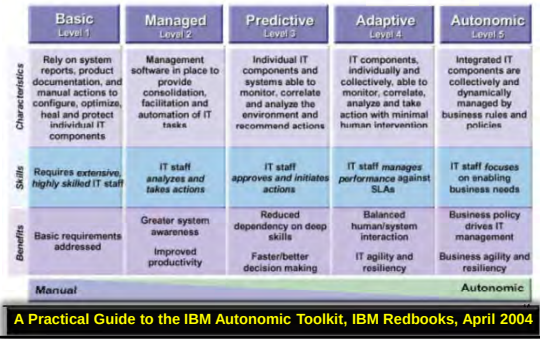
14

Levels of Autonomic Computing Maturity



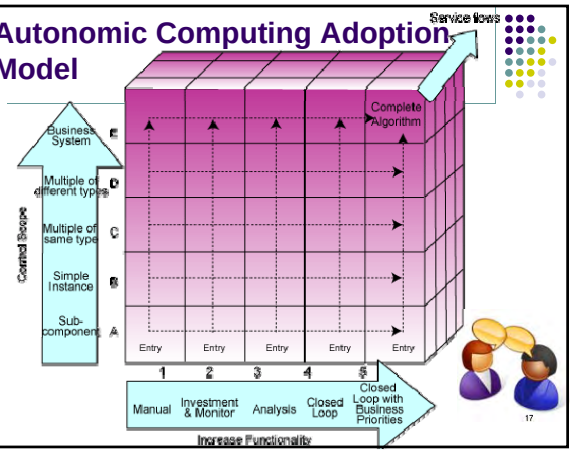
A Practical Guide to the IBM Autonomic Toolkit, IBM Redbooks, April 2004

Levels of Autonomic Computing Maturity



A Practical Guide to the IBM Autonomic Toolkit, IBM Redbooks, April 2004

Autonomic Computing Adoption Model



17