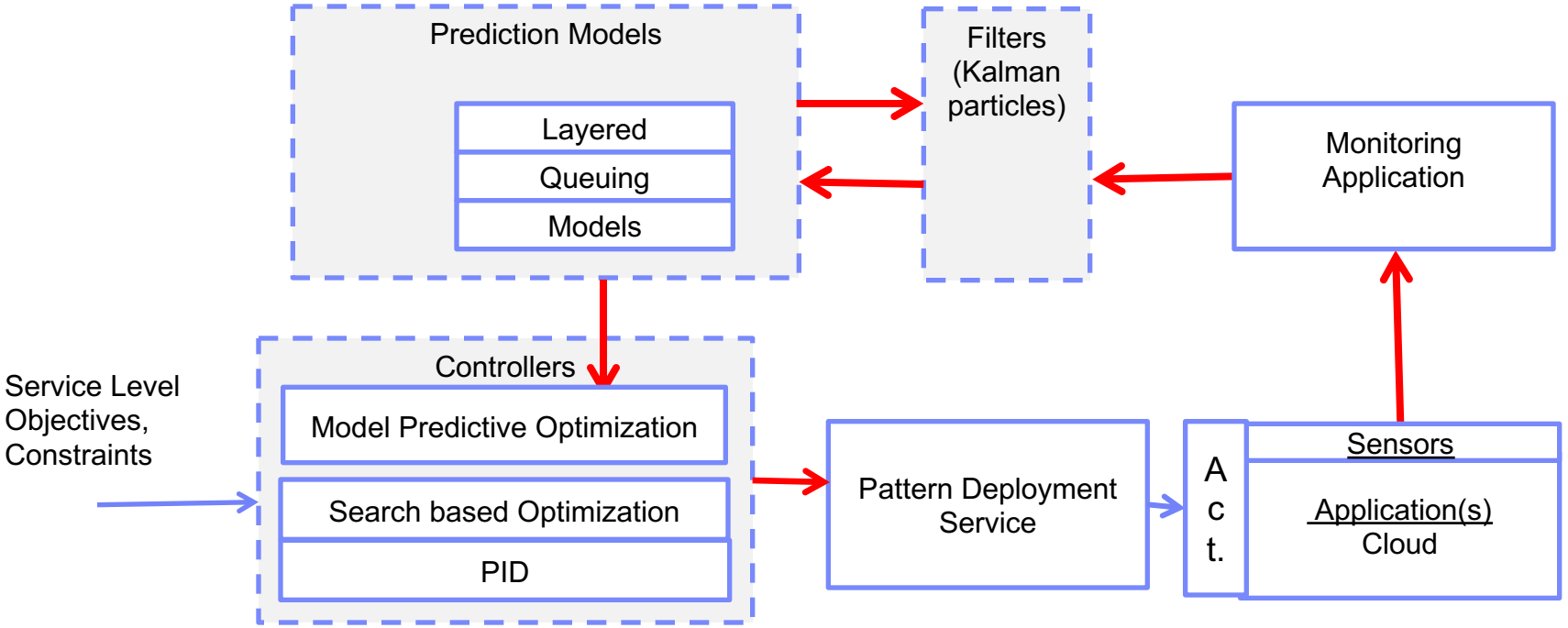
YORK UNIVERSITÉ UNIVERSITY

ASRL
Adaptive Systems Research Lab

Engineering Adaptive Software Systems: Policies

Marin Litoiu
Adaptive Systems Research Lab
York University
Department of EECS and School of IT
http://ceraslabs.com

Look Ahead Adaptation

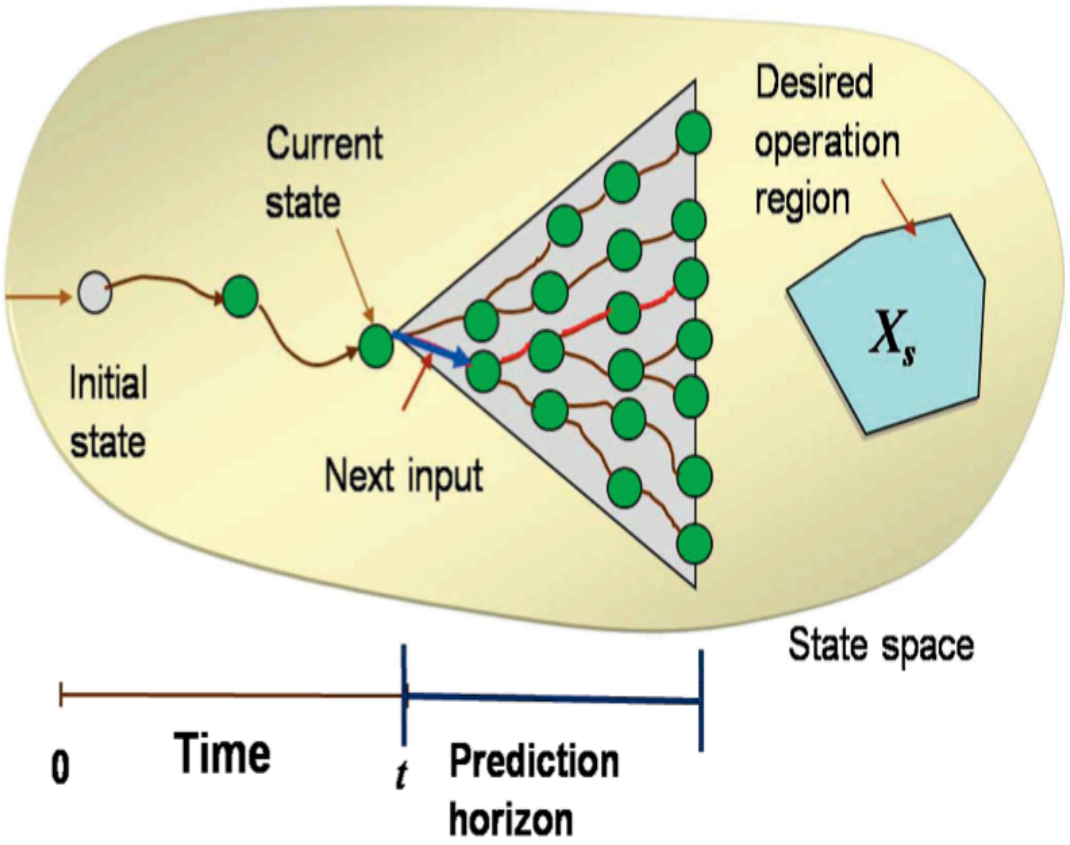


- **Predictive (red arrows): anticipates future load, performance, cost**
  - Uses prediction models, filters and predictive optimization. It is slow and effortful but efficient

Look Ahead Optimization with Thrashing in Cloud

- **The problem:**
  - Given a set of applications in a private cloud, with variable workloads
    - Minimize the cost, performance, trashing
      - By allocating/migrating VMs to physical machines in cloud
- **How it works:**
  - Monitor the current states
  - Predict the workload over a look ahead horizon T
  - Compute the allocation for each step t=1, 2…T
    - For each step, use the model of the system to predict the cost
  - Implement the allocation for step 1
  - Repeat Monitor, Predict, Compute, Implement steps….

Look Ahead Adaptation



From [Bai and Abdelwahed]

Self-Managing Policies

- **Autonomic elements function at different levels of abstraction**
- **At the lowest levels, the capabilities and the interaction range of an autonomic element are limited and hard-coded**
- **At higher levels, elements pursue more flexible goals specified with policies, and the relationships among elements are flexible and may evolve over time**
- **Action, goal and utility-function policies**

Policy Examples

- **A policy is a set of considerations designed to guide decisions of courses of action.**
- **"Neither a borrower, nor a lender be; for a loan oft loses both itself and friend, and borrowing dulls the edge of husbandry." In *Hamlet*, Shakespeare's policy regarding borrowing.**
- **Star Wars**
  - When C3PO, upon receiving caution from Hans Solo, tells R2D2 to "let the wookie win." Apparently Chewbacca (the wookie in question) had a habit of detaching an opponent's arm upon losing.
  - It is important to note that R2D2 had another implicit policy that said when he's competing, he should try to win, and this policy directly conflicted with Solo's sage advice.
  - In the end, R2D2 let the Wookie have the game, valuing his arm over the victory.

Action Policies

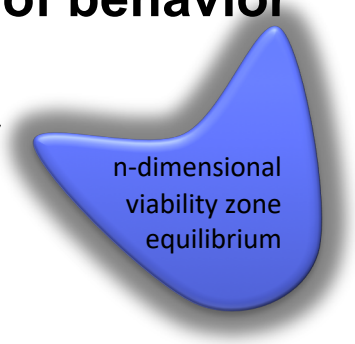
- **Dictate the actions that should be taken when the system is in a given state**
- **IF (condition) THEN (action)**
  - where the condition specifies either a specific state or a set of possible states that all satisfy the given condition
- **Note that the state that will be reached by taking the given action is not specified explicitly**
- **Policy author knows which state will be reached upon taking the recommended action and deems this state more desirable than states that would be reached via alternative actions**

Goal Policies

- **Rather than specifying exactly what to do in the current state, goal policies specify either a single desired state, or one or more criteria that characterize an entire set of desired states**
- **Rather than relying on a human to explicitly encode rational behavior, as in action policies, the system generates rational behavior itself from the goal policy**
- **This type of policy permits greater flexibility and frees human policy makers from the "need to know" low-level details of system function, at the cost of requiring reasonably sophisticated planning or modeling algorithms**

Utility-Function Policies

- An objective function that expresses the value of each possible state
- Generalized goal policies
- Instead of performing a binary classification into desirable versus undesirable states, they ascribe a real-valued scalar desirability to each state
- Because the most desired state is not specified in advance, it is computed on a recurrent basis by selecting the state that has the highest utility from the present collection of feasible states
- Provide more fine-grained and flexible specification of behavior than goal and action policies
- Allow for unambiguous, rational decision making by specifying the appropriate tradeoff
- Preferences are difficult to elicit and specify



Autonomic Computing Policies

- **What is the difference between action, goal and utility-function policies?**
  - Advantages, disadvantages, benefits, limitations?

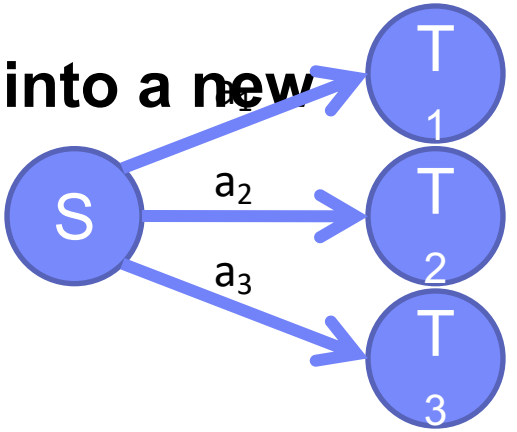
Policy Types for Autonomic Computing

- **Action policies**
  - **If-then action rules** specify exactly what to do under the current condition.
  - Rational behaviour is compiled in by the designer
- **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.

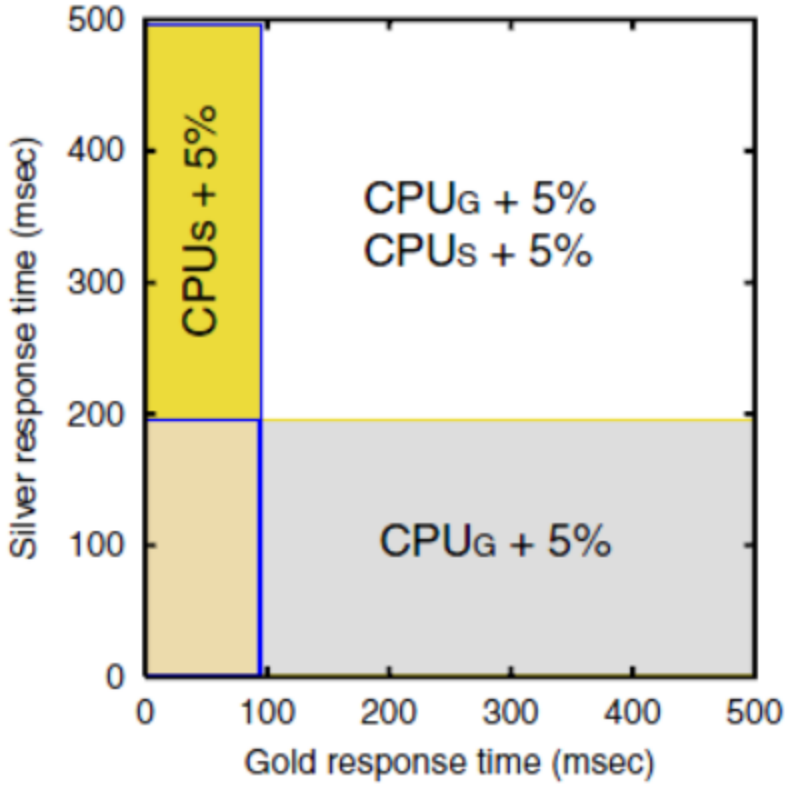
Action Policies

- **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***



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%



Action Policy Example

- **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
- **Condition**
  - None
- **Action**
  - Block any new user logins

Goal Policies

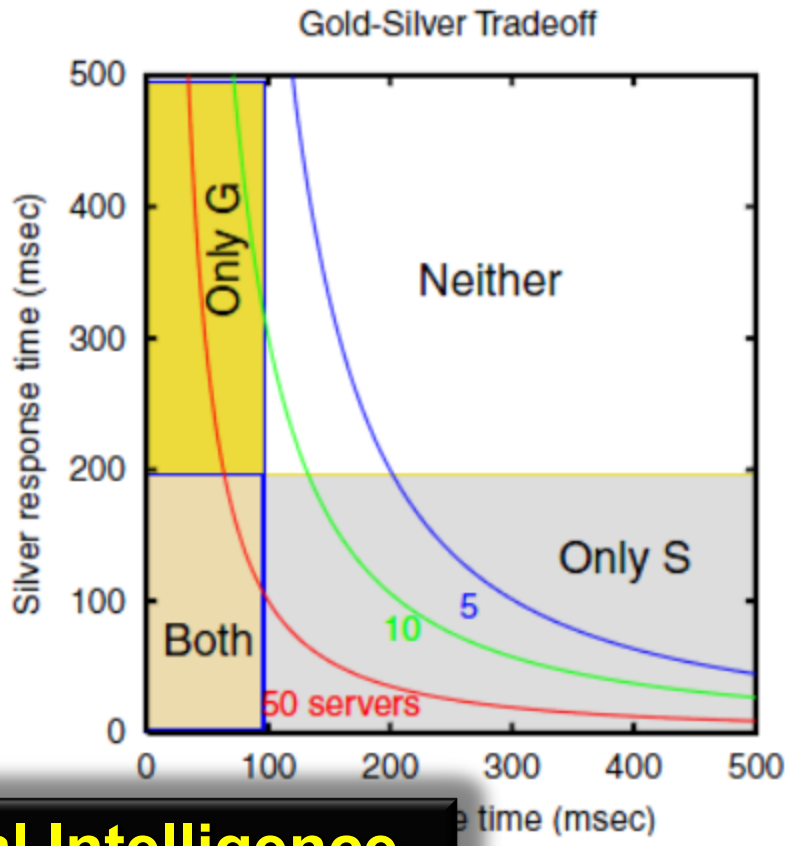
- **Instead of specifying what to do in a current state, specify a single desired state or a set of desired states are**
- **Any member of this target set is equally acceptable**
- **Cannot express fine distinctions in preference**
- **How to compute a set of actions that gets the system from current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

Goal Policy Example

- **RT: Response Time**

  Gold: RT_{Gold} <= 100 ms

  Silver: RT_{Silver} <= 200 ms



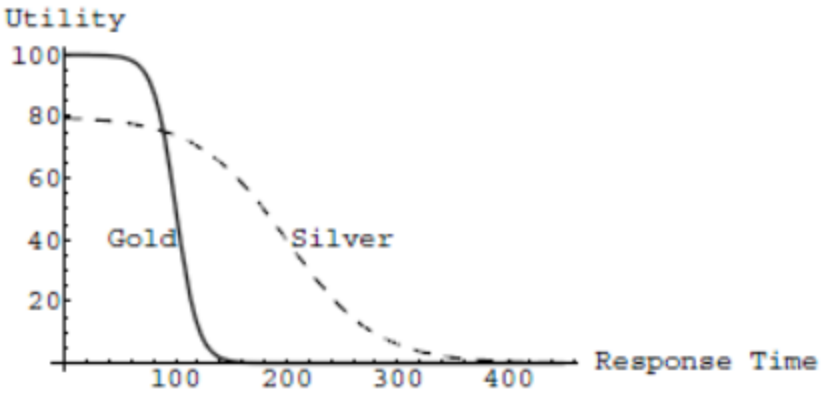
Kephart and Walsh: An Artificial Intelligence Perspective on Autonomic Computing Policies, POLICY

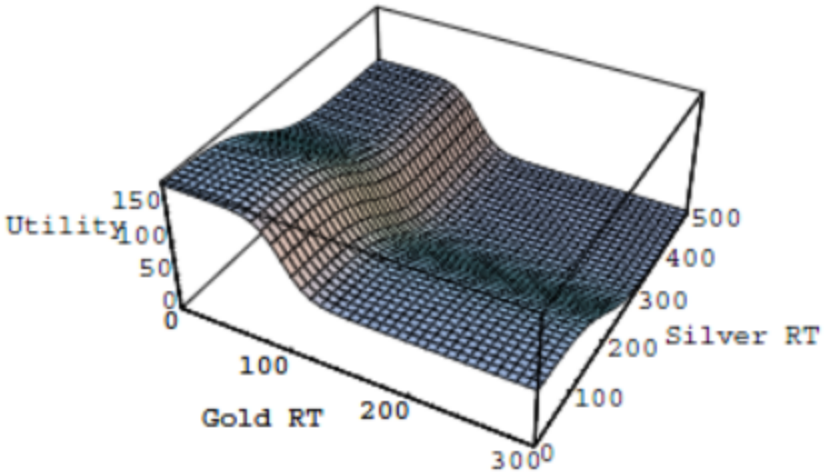
Utility-Function Policies

- **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**
- **Unambiguous, rational decision making**

Utility-Function Policy Example

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





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**

Policy Languages

- **PDL**
- **Ponder**
- **Rei**
- **KAoS**
- **ACPL, SPL (PMAC)**
- **XACML**
- **CIM-SPL**

Ponder

- **Developed @ Imperial College London**
  - A declarative, object-oriented language for specifying different types of policies
  - grouping policies into roles and relationships
  - defining configurations of roles and relationships as management structures.
  - Can be used to specify security policies with role-based access control, as well as general purpose management policies

More info

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

TOWARDS A SE METHODOLOGY

Engineering Adaptive Loops: Methodology*

*Parisa Zoghi, Mark Shtern, Marin Litoiu, and Hamoun Ghanbari. 2016. Designing Adaptive Applications Deployed on Cloud Environments. *ACM Trans. Auton. Adapt. Syst.* 10, 4, Article 25

- **Elicitation phase (Goals and operations)**
- **Ranking phase**
- **Adaptation algorithm phase**

1. Adaptation Goals

2. Control Points

- **Definition**
  - Operations that called at run-time cause controlled changes in a system
  - Can belong to services, applications, cloud
- **Examples**
  - Add/remove instances
  - Launch instances in cheaper availability zone

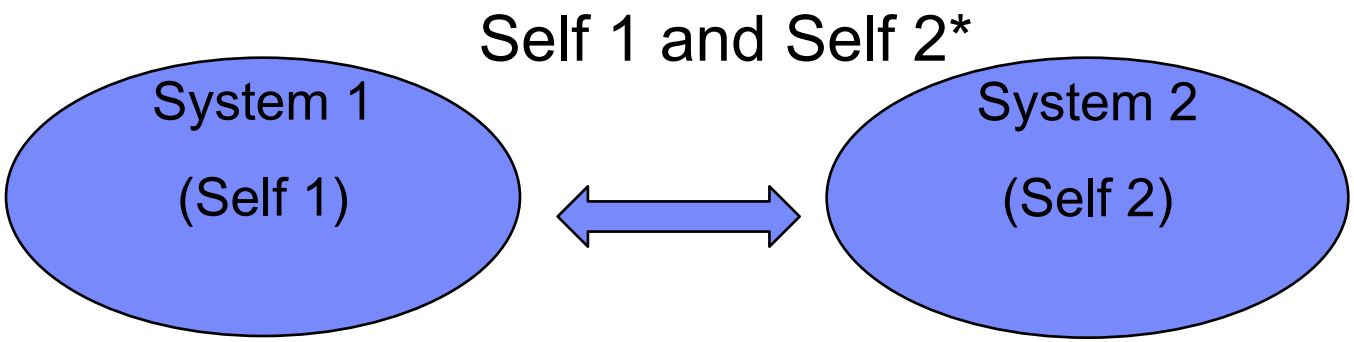
3. Ranking Phase

Control point alternatives	Numerical values
Alternative 1	1
Alternative 2	2
...	...
Alternative n	n

4. Implement MAPE Loops

- **Monitor** **Goals**
- **Analyze** **and** **Plan**
  - Compute a Strategy over the space of Control Points using a Model
    - MPC
    - LQR
    - Search Based Algorithms
- **Execute** the **Strategy**

Conclusions



- **Intuitive**
- **Reactive**
- **Fast**
- **Low resource consumption**
- **Unconscious**
- **Useful but not always right**
- **Ex: 2 x 9=?**

- **Analytical**
- **Deliberative**
- **Slow**
- **High resource consumption**
- **Conscious**
- **More often right**
- **Ex: 69 x 37 =?**

Reactive Adaptation ⟺ Model Based Adaptation

*Kahneman, Thinking fast and slow