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

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

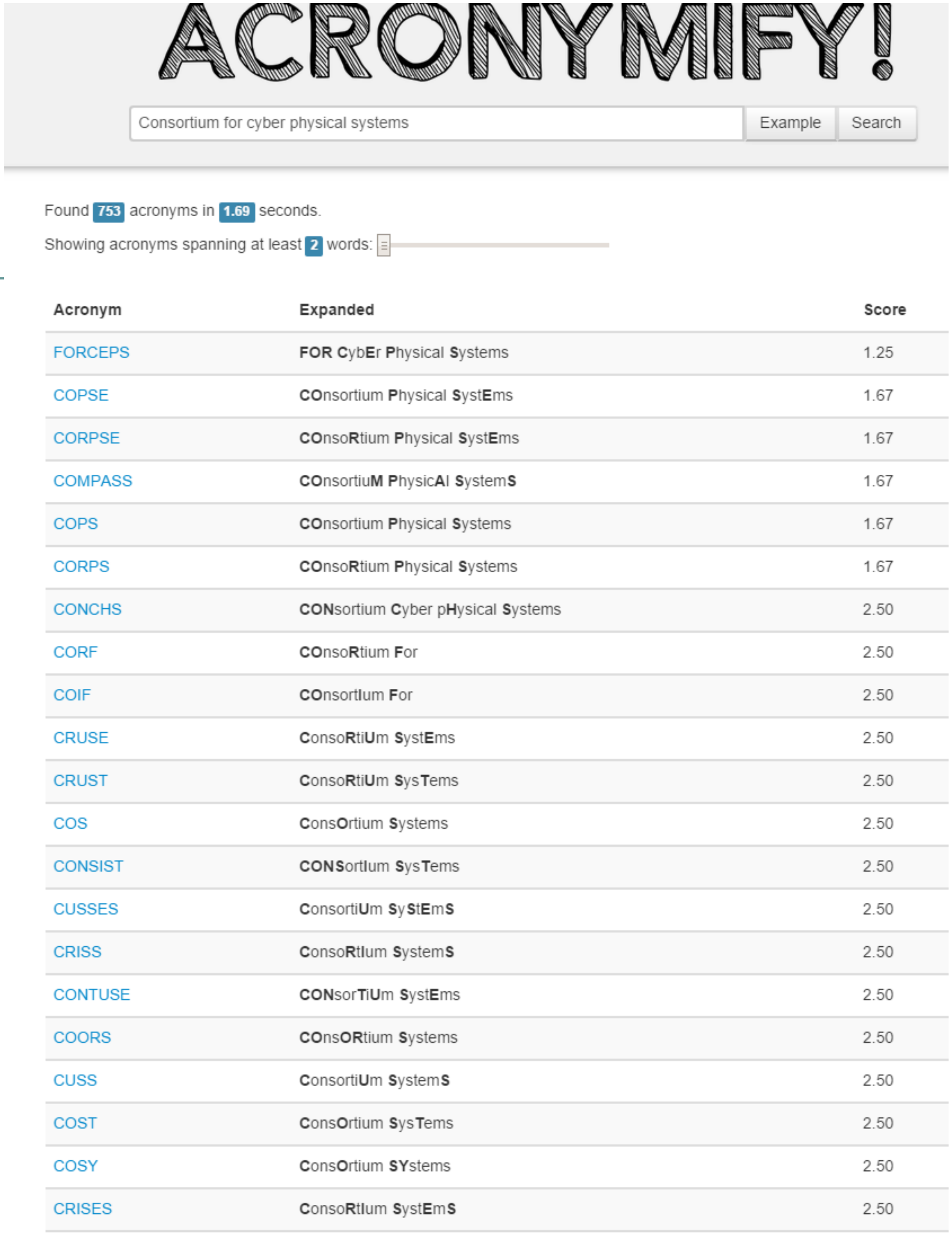
http://courses.seng.uvic.ca/courses/2015/summer/seng/480a
http://courses.seng.uvic.ca/courses/2015/summer/csc/485a
http://courses.seng.uvic.ca/courses/2015/summer/csc/586a

# Announcements

- A3
  - Posted and due July 10
- Grad project
  - Handed out June 24
  - Due July 25
- Marks posted
  - A1 — Avg: 86.8; Med: 88
  - M1 — Avg: 78.9; Med: 86

# Needed your help

- Catchy acronym for grant proposal **Consortium for Cyber Physical Systems Research**
- Engr Dean won with — CYPHY
- Simon Taft was close: http://acronymify.com



Consortium for cyber physical systems

Found 753 acronyms in 1.69 seconds.

Showing acronyms spanning at least 2 words:

| Acronym | Expanded | Score |
|---|---|---|
| FORCEPS | FOR CybEr Physical Systems | 1.25 |
| COPSE | COnsortium Physical SystEms | 1.67 |
| CORPSE | COnsoRtium Physical SystEms | 1.67 |
| COMPASS | COnsortiuM PhysicAl SystemS | 1.67 |
| COPS | COnsortium Physical Systems | 1.67 |
| CORPS | COnsoRtium Physical Systems | 1.67 |
| CONCHS | CONsortium Cyber pHysical Systems | 2.50 |
| CORF | COnsoRtium For | 2.50 |
| COIF | COnsortIum For | 2.50 |
| CRUSE | ConsoRtiUm SystEms | 2.50 |
| CRUST | ConsoRtiUm SysTems | 2.50 |
| COS | ConsOrtium Systems | 2.50 |
| CONSIST | CONSortIum SysTems | 2.50 |
| CUSSES | ConsortiUm SyStEmS | 2.50 |
| CRISS | ConsoRtIum SystemS | 2.50 |
| CONTUSE | CONsorTiUm SystEms | 2.50 |
| COORS | COnsORtium Systems | 2.50 |
| CUSS | ConsortiUm SystemS | 2.50 |
| COST | ConsOrtium SysTems | 2.50 |
| COSY | ConsOrtium SYstems | 2.50 |
| CRISES | ConsoRtIum SystEmS | 2.50 |

# Autonomic Computing Vision

Autonomic Computing is really about making systems self-managing …

—Paul Horn, IBM Research, 2001

# New Reading Assignment

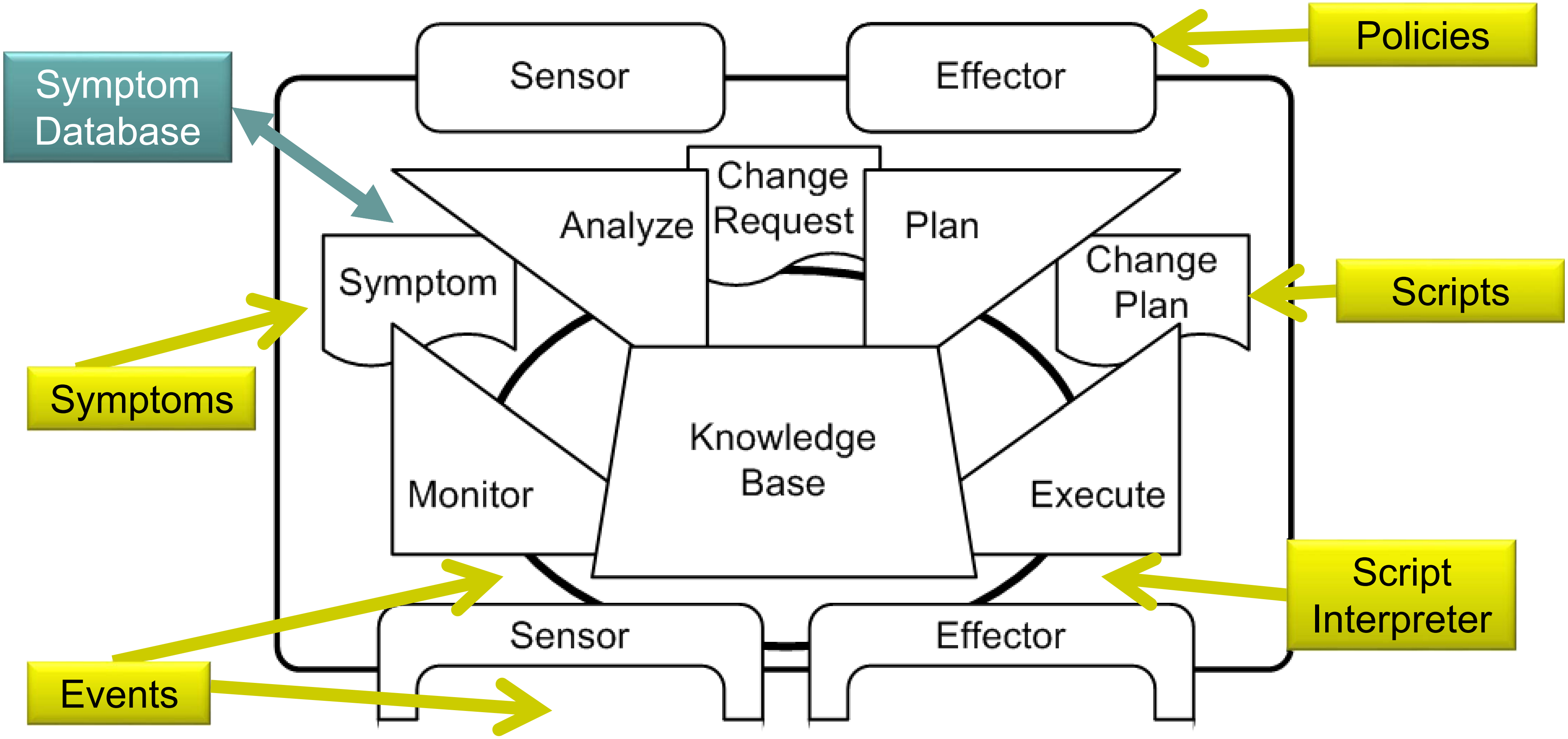
- Kephart & Walsh; An AI Perspective on AC Policies, 5th IEEE International Workshop on Policies for Distributed Systems and Networks (Policy 2004) https://www.researchgate.net/publication/220726111_An_Artificial_Intelligence_Perspective_on_Autonomic_Computing_Policies http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1309145&punumber%3D9150%26filter%3DAND%28p_IS_Number%3A29053%29
- John Wilkes, HP Labs: Utility functions, prices, and negotiation, HP Tech Report and Slides (2008) http://www.e-wilkes.com/john/papers/HPL-2008-81.pdf

# Implementing Autonomic Elements

**The devil lies in the details ...**

**Standards, data and control integration, interfaces, endpoints, services, SOA ...**

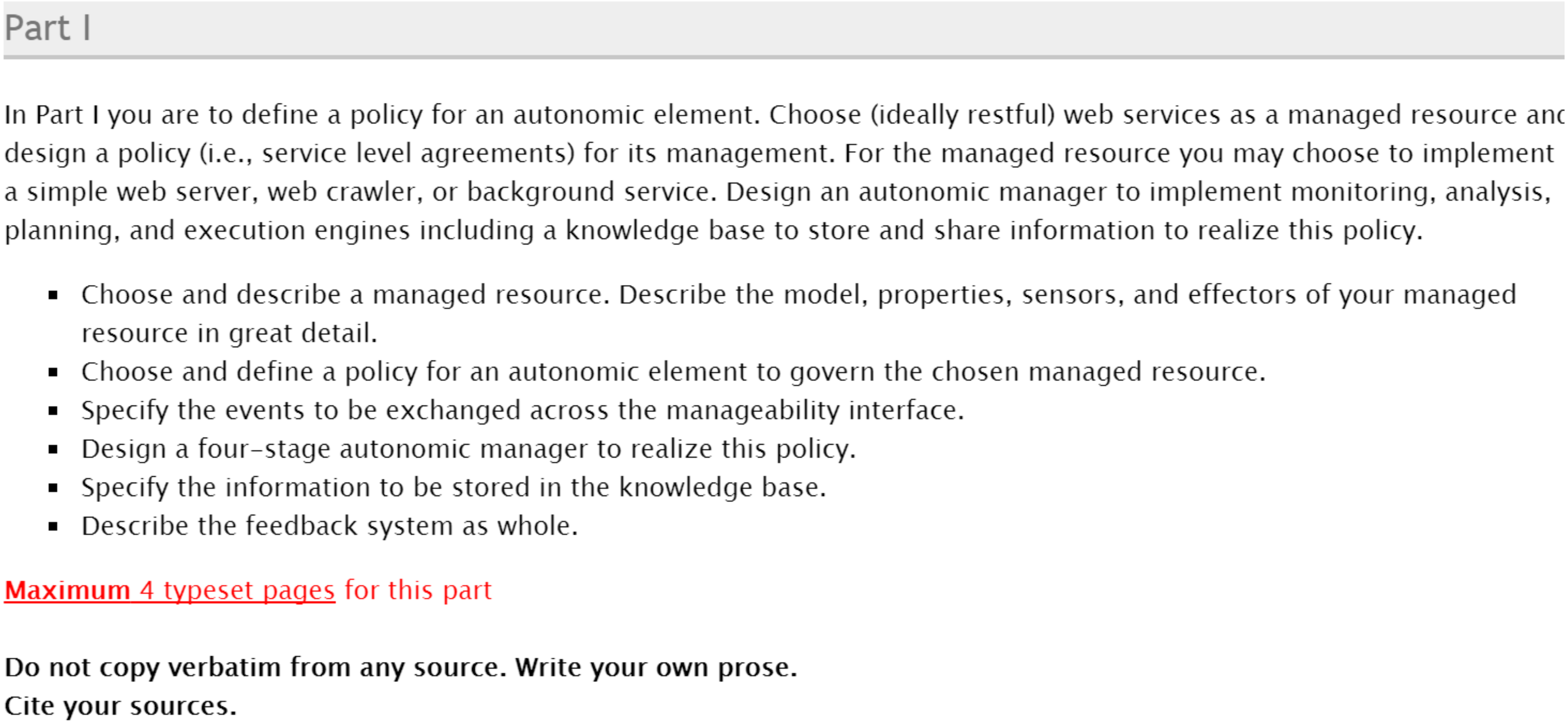
# MAPE-K Loop
# Standards & Interfaces



# Assignment 3

- Two parts
  - Design and implement an autonomic element
  - Control restful web services
  - Part I Design (individual)
  - Part II Implementation (group)
- Resources
  - OASIS: Web Services Distributed Management: Management of Web Services (WSDM-MOWS) 1.1 OASIS Standard (2006)
  - OASIS: Web Services Distributed Management: Management Using Web Services (WSDM-MUWS) 1.1 OASIS Standard (2006)
  - Kreger, H., Studwell, T.: Autonomic Computing and Web Services Distributed Management (2005)

# Assignment 3 — Part I



## Part I

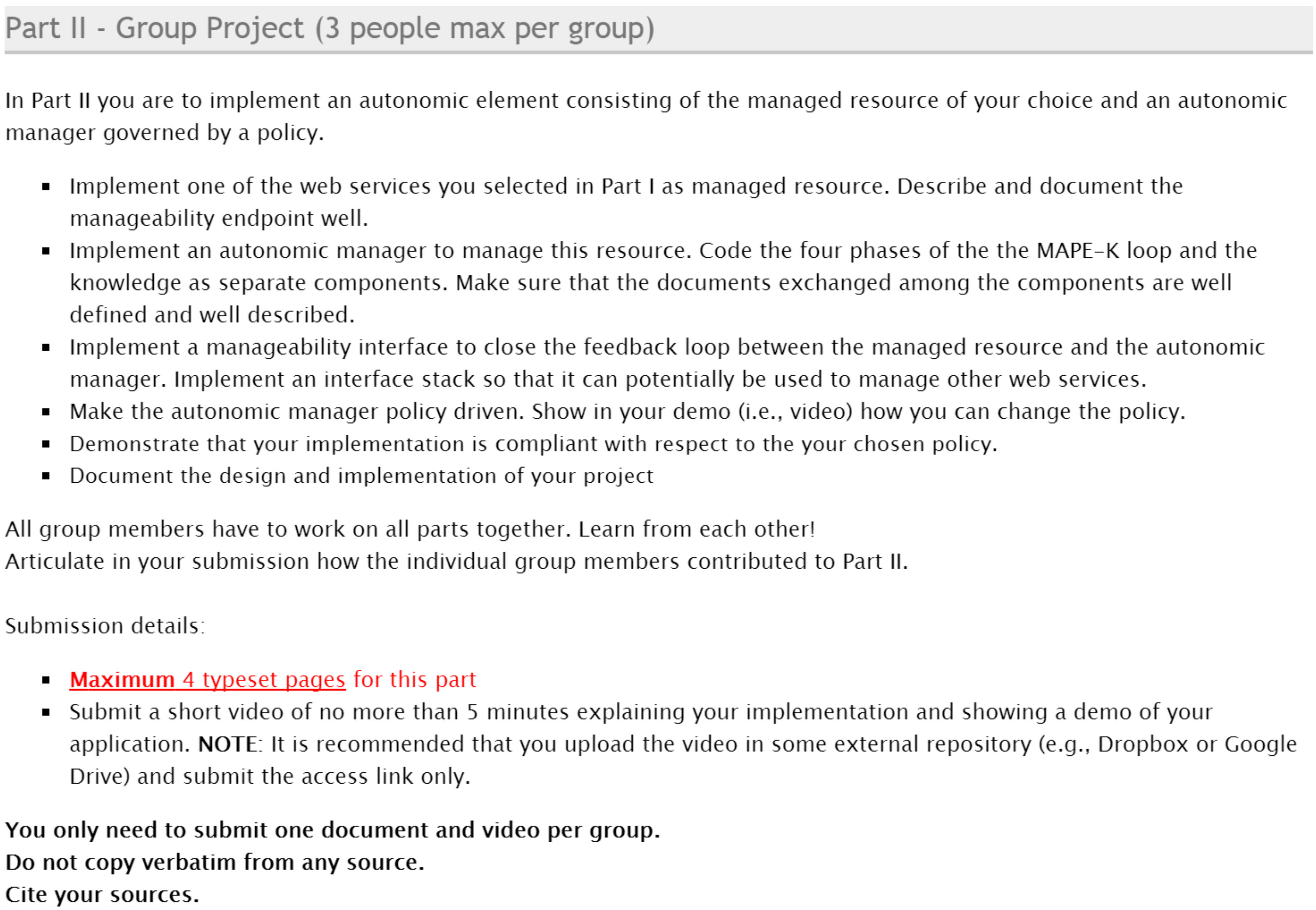
In Part I you are to define a policy for an autonomic element. Choose (ideally restful) web services as a managed resource anc design a policy (i.e., service level agreements) for its management. For the managed resource you may choose to implement a simple web server, web crawler, or background service. Design an autonomic manager to implement monitoring, analysis, planning, and execution engines including a knowledge base to store and share information to realize this policy.

- Choose and describe a managed resource. Describe the model, properties, sensors, and effectors of your managed resource in great detail.
- Choose and define a policy for an autonomic element to govern the chosen managed resource.
- Specify the events to be exchanged across the manageability interface.
- Design a four-stage autonomic manager to realize this policy.
- Specify the information to be stored in the knowledge base.
- Describe the feedback system as whole.

**Maximum 4 typeset pages** for this part

**Do not copy verbatim from any source. Write your own prose.**
**Cite your sources.**

# Assignment 3 — Part I



## Part II - Group Project (3 people max per group)

In Part II you are to implement an autonomic element consisting of the managed resource of your choice and an autonomic manager governed by a policy.

- Implement one of the web services you selected in Part I as managed resource. Describe and document the manageability endpoint well.
- Implement an autonomic manager to manage this resource. Code the four phases of the the MAPE-K loop and the knowledge as separate components. Make sure that the documents exchanged among the components are well defined and well described.
- Implement a manageability interface to close the feedback loop between the managed resource and the autonomic manager. Implement an interface stack so that it can potentially be used to manage other web services.
- Make the autonomic manager policy driven. Show in your demo (i.e., video) how you can change the policy.
- Demonstrate that your implementation is compliant with respect to the your chosen policy.
- Document the design and implementation of your project

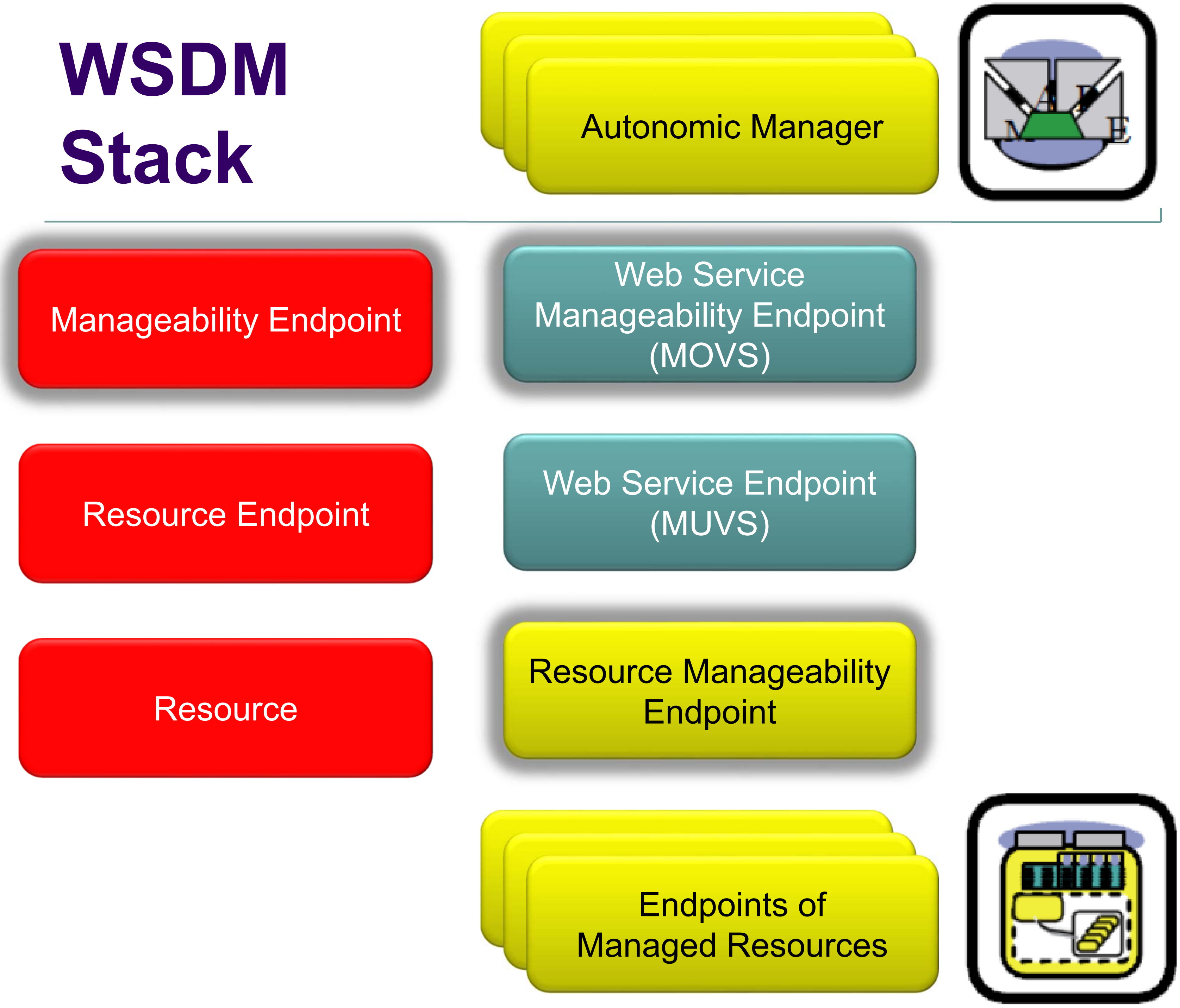
All group members have to work on all parts together. Learn from each other!
Articulate in your submission how the individual group members contributed to Part II.

Submission details:

- **Maximum** 4 typeset pages for this part
- Submit a short video of no more than 5 minutes explaining your implementation and showing a demo of your application. **NOTE**: It is recommended that you upload the video in some external repository (e.g., Dropbox or Google Drive) and submit the access link only.

**You only need to submit one document and video per group.**
**Do not copy verbatim from any source.**
**Cite your sources.**

# WSDM Stack



Autonomic Manager

Manageability Endpoint

Web Service Manageability Endpoint (MOVS)

Resource Endpoint

Web Service Endpoint (MUVS)

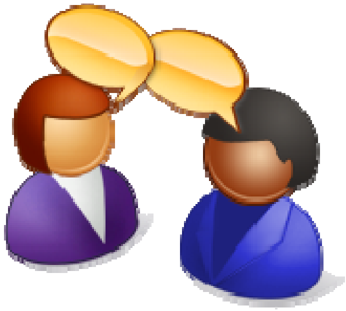
Resource

Resource Manageability Endpoint

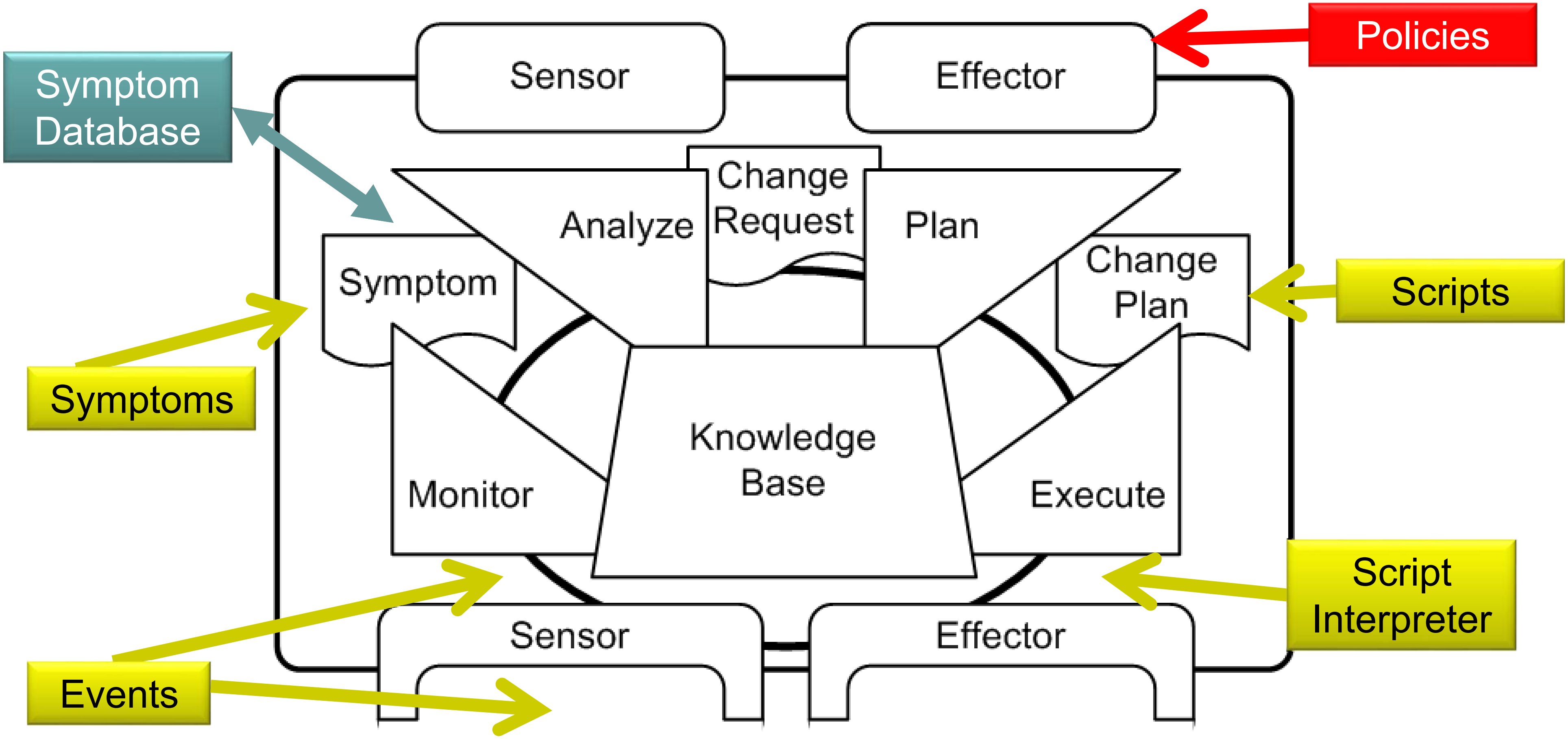
Endpoints of Managed Resources

# Autonomic Element Architecture Assignment 3

| | | | |
|---|---|---|---|
| | Sample Autonomic Manager | | Implementation Autonomic Manager |
| | Interface Autonomic Manager | | Interface Autonomic Manager |
| Event Persistence | Event | Sample Main | Enterprise Event Bus or WSDM with CBE |
| | Interface Managed Element | | Interface Managed Element |
| | Touchpoint Managed Element | | Endpoint Managed Element |
| | Sample Managed Element | | Implementation Managed Element |



# MAPE-K Loop
# Standards & Interfaces



# 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

**Kephart & Walsh; An AI Perspective on AC Policies, 5th IEEE International Workshop on Policies for Distributed Systems and Networks (Policy 2004)**

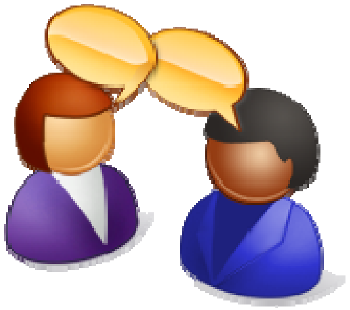
# Policy Examples

- 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.

**D. Kaminsky, IBM Software Architect**
**An Introduction to Policy for Autonomic Computing, 2005.**

# Autonomic Computing Policies

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



# 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

n-dimensional
viability zone
equilibrium

**Kephart & Walsh; An AI Perspective on AC Policies, 5th IEEE International Workshop on Policies for Distributed Systems and Networks (Policy 2004)**

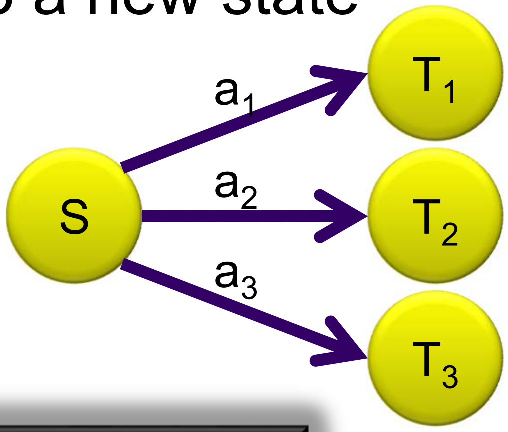
# Policy Types for Autonomic Computing

- 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.

# Action Policies

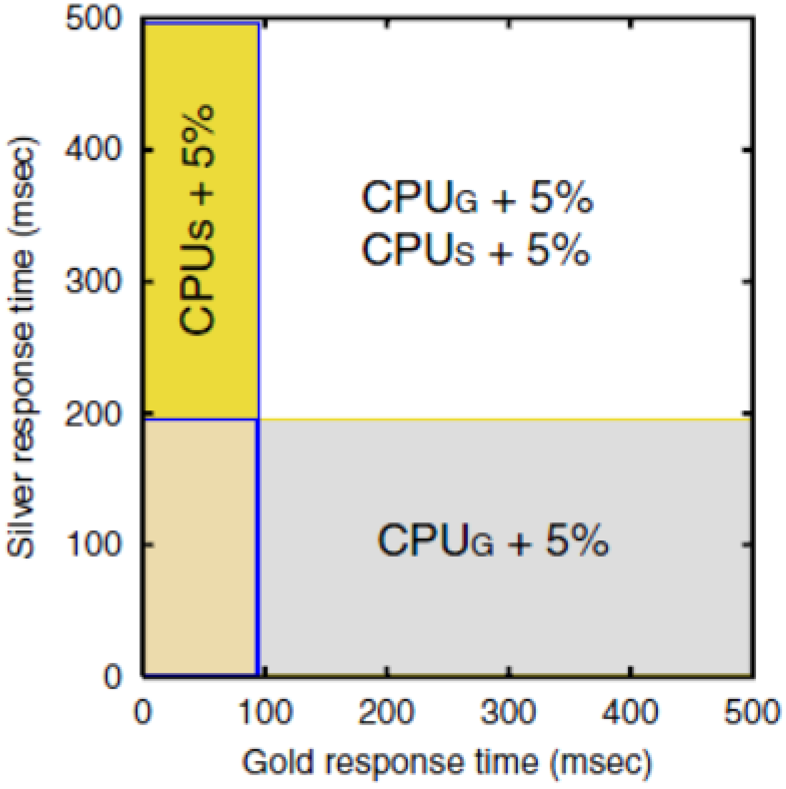
- A state *S* is a vector of attributes
- *S* can directly be measured by a sensor, or
- *S* can be inferred or synthesized from lower-level measurements
- Policy will directly or indirectly cause an action *a*
- Deterministic or probabilistic transition into a new state from *S* to a new state *T*



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

# 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 Examples

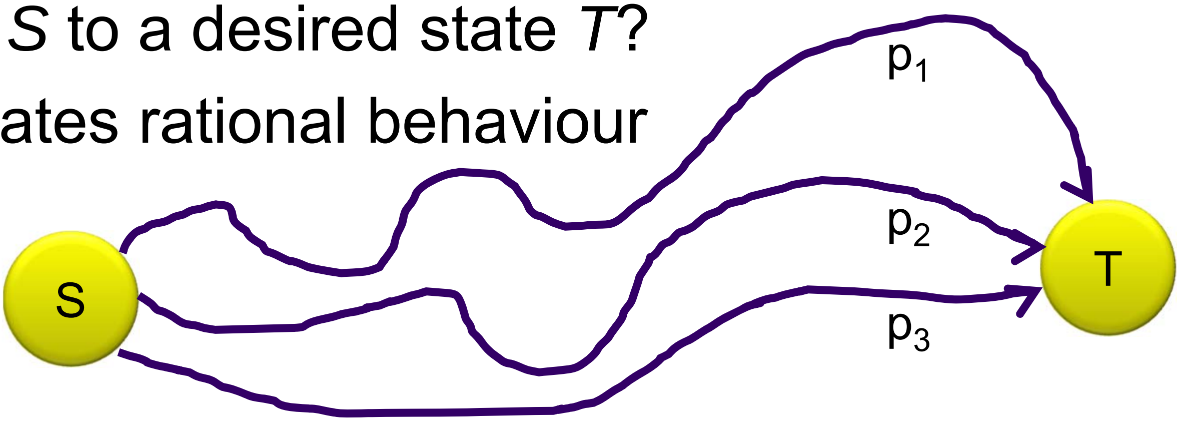
- For each machine, if idle session is greater than 20 minutes then terminate the session
- BitTorrent user processes initiated from IP address 141.223.2.15 should have lowest priority

```
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

# Goal Policies

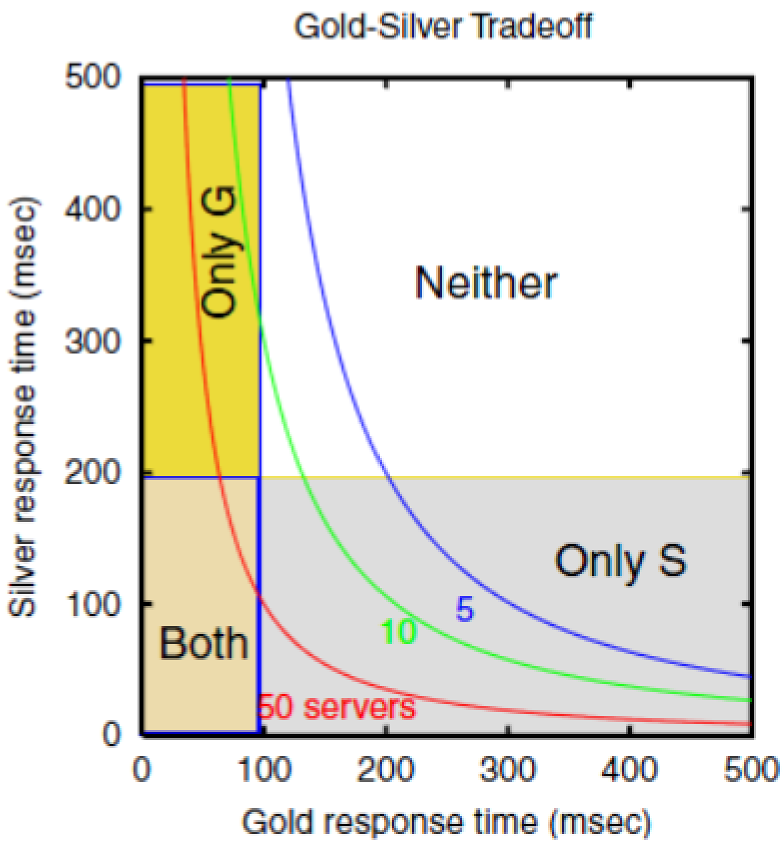
- Instead of specifying what to do in a current state, specify a single desired state or a set of desired states
- Any member of this target set is equally acceptable
- Cannot express fine distinctions in preference
- How to compute a set of actions that gets the system from current state *S* to a desired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

# 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.**

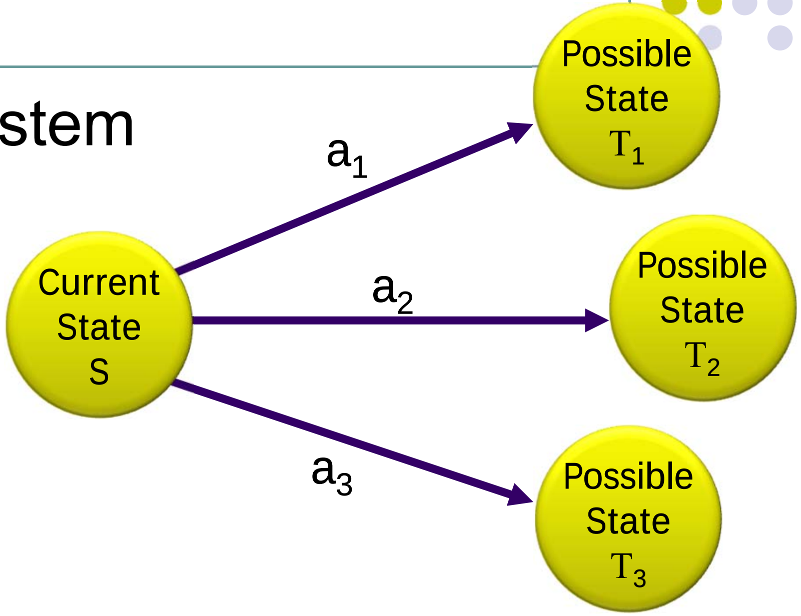
# Utility-Function Policies

- 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.**

# 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



Current State S

$a_1$

Possible State $T_1$

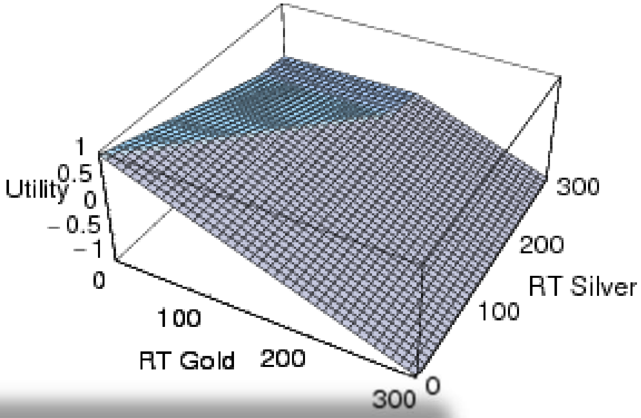
$a_2$

Possible State $T_2$

$a_3$

Possible State $T_3$

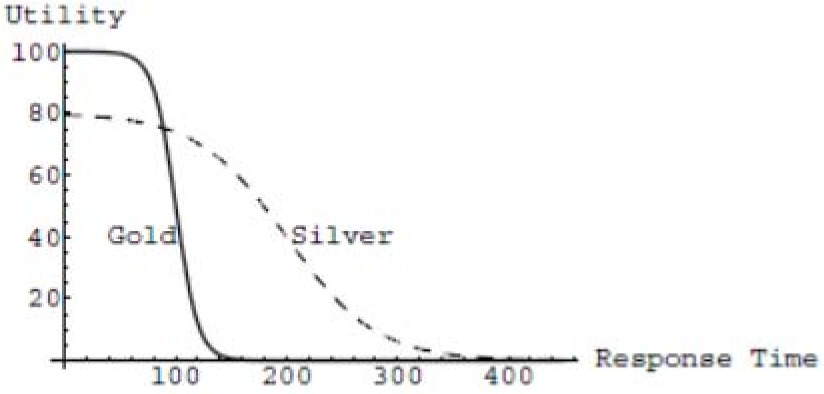
U(**RT**) =

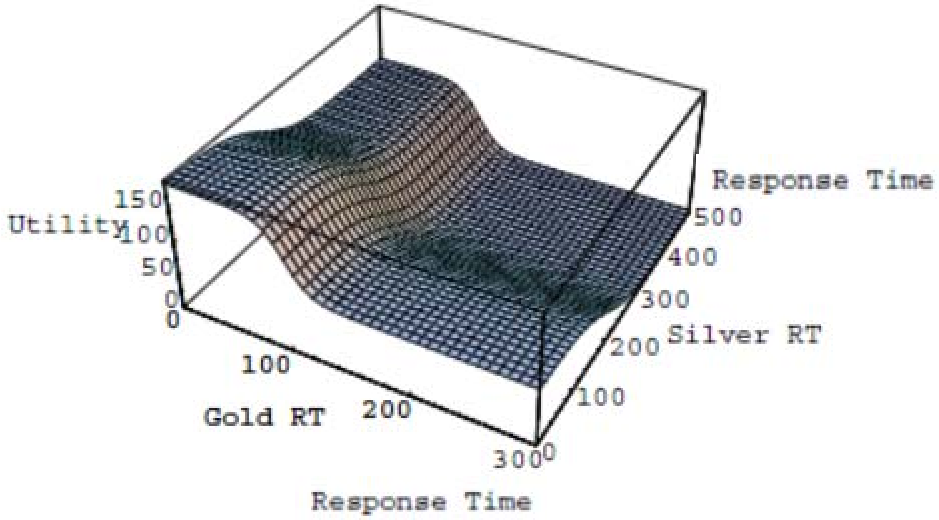


**Kephart and Walsh: An Artificial Intelligence Perspective on Autonomic Computing Policies, POLICY 2004.**

# 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.**

# Policy Types for Autonomic Computing

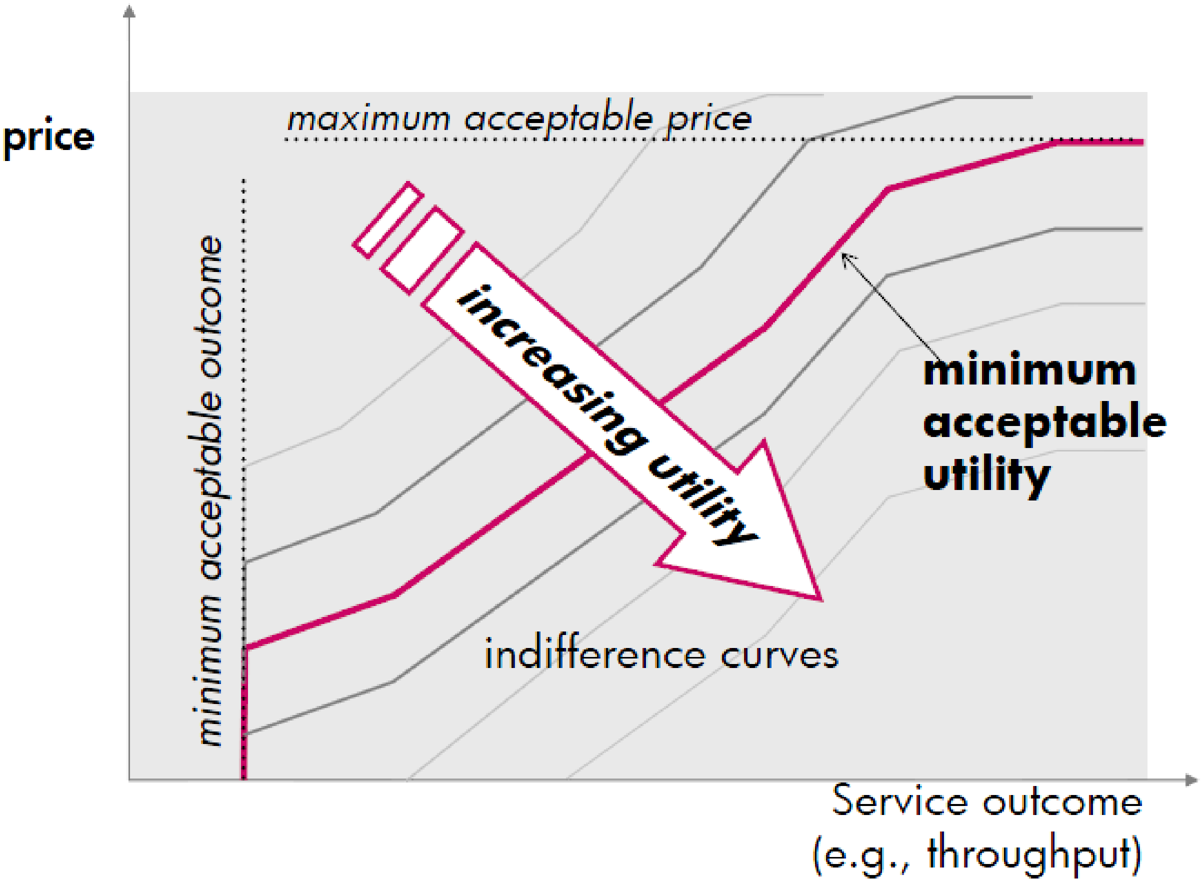
- Real autonomic systems embody a combination of policy types
- In ACRA
  - Lower levels typically use action policies
  - For higher levels, goal or utility-function policies are more appropriate
- Unified framework is needed to support multiple policy types within a single autonomic component

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

# Policy Model and Terminology

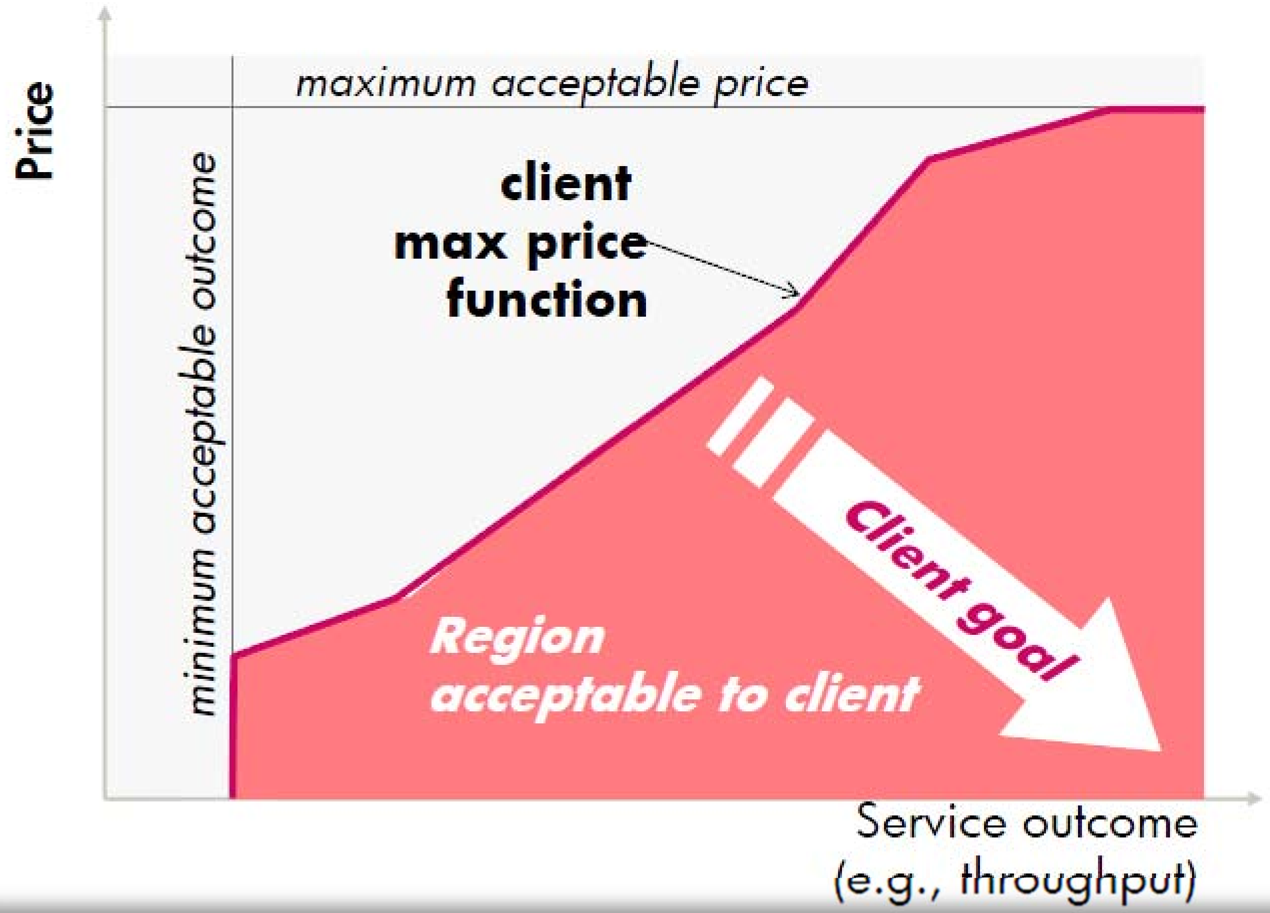
- **Policy**
  - A collection of policy alternatives.
- **Policy Alternative**
  - A collection of policy assertions.
- **Policy Assertion**
  - Represents an individual requirement, capability or other property of a behavior.
- **Policy Assertion Type**
  - Represents a class of policy assertions and 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.

# Minimal Acceptable Utility


price
maximum acceptable price
minimum acceptable outcome
increasing utility
minimum acceptable utility
indifference curves
Service outcome (e.g., throughput)

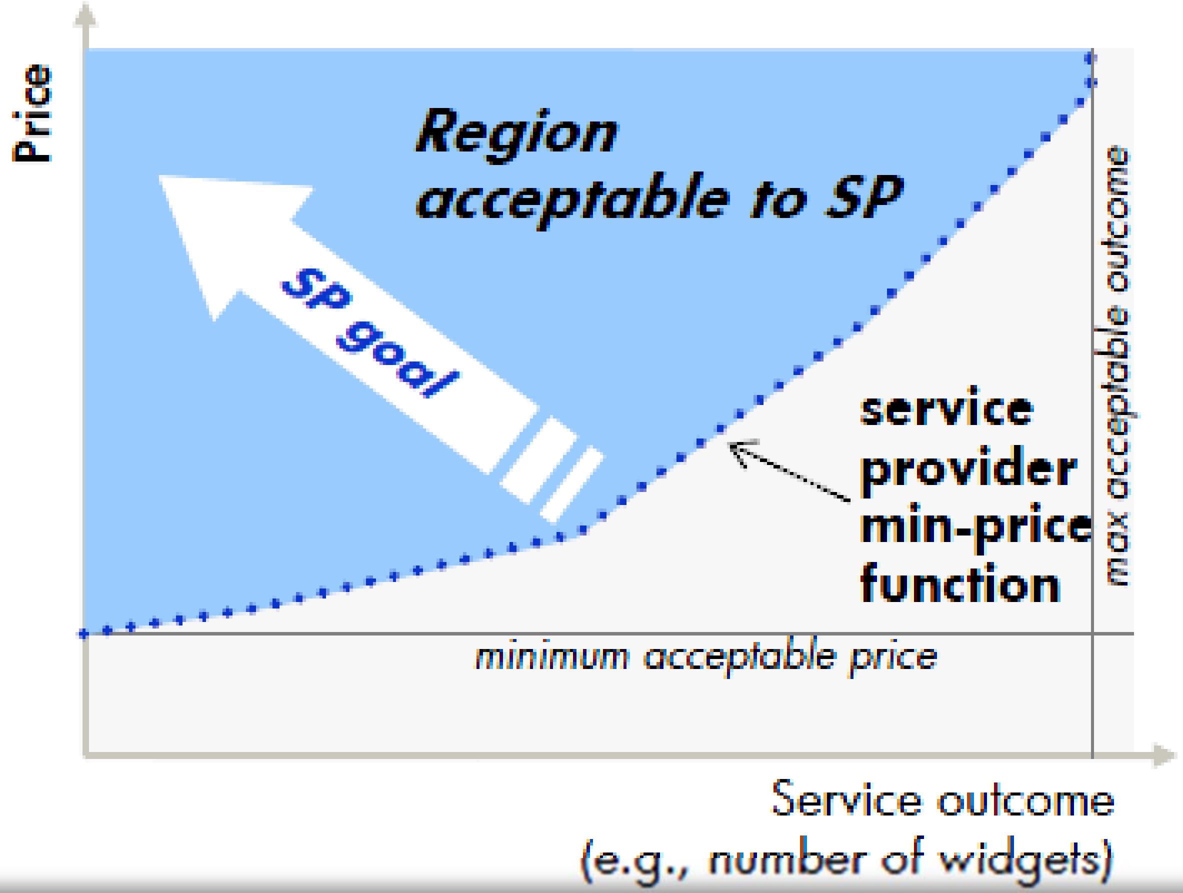
**John Wilkes, HP Labs: Utility functions, prices, and negotiation, HP Tech Report and Slides, 2008.**

# Client Utility



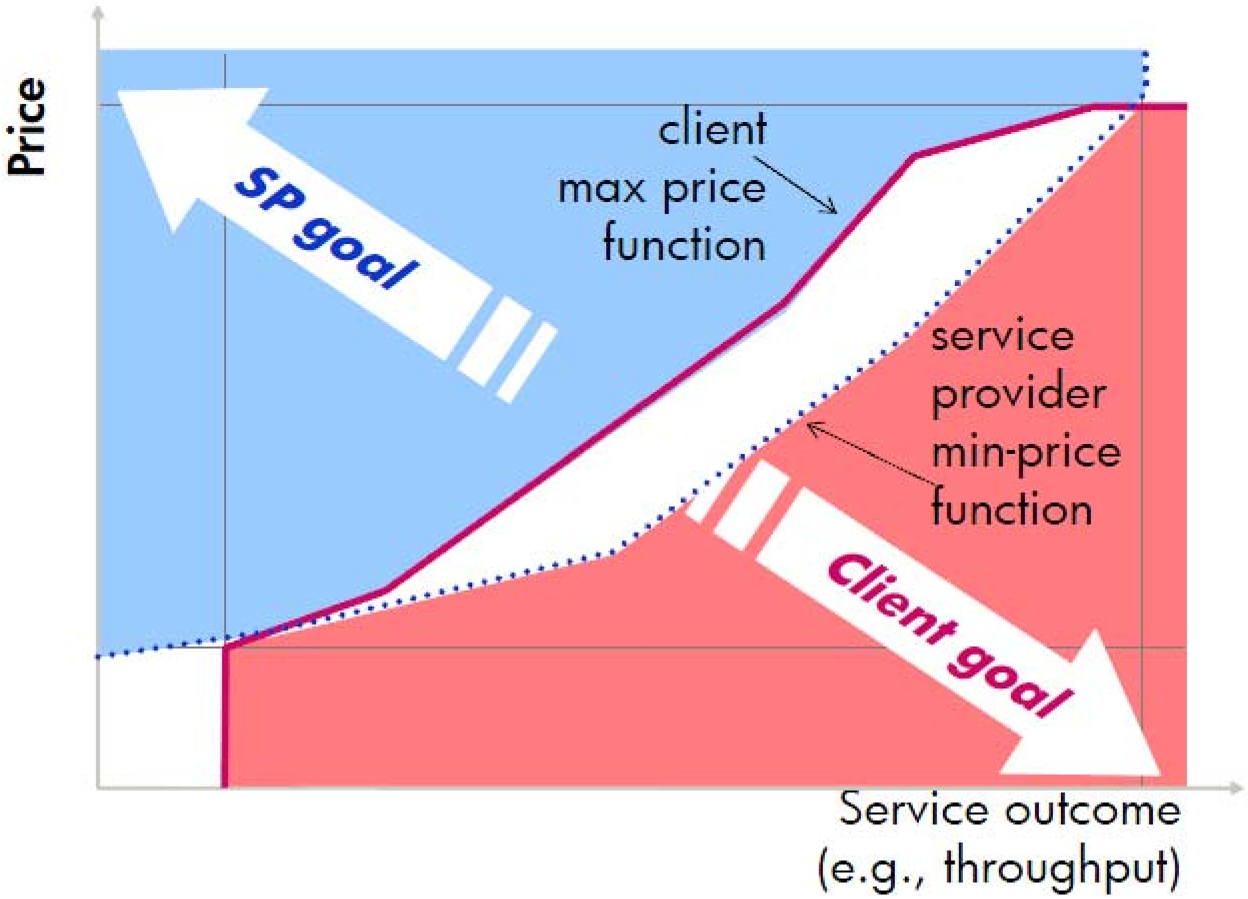
**John Wilkes, HP Labs: Utility functions, prices, and negotiation, HP Tech Report and Slides, 2008.**

# Service Provider Utility



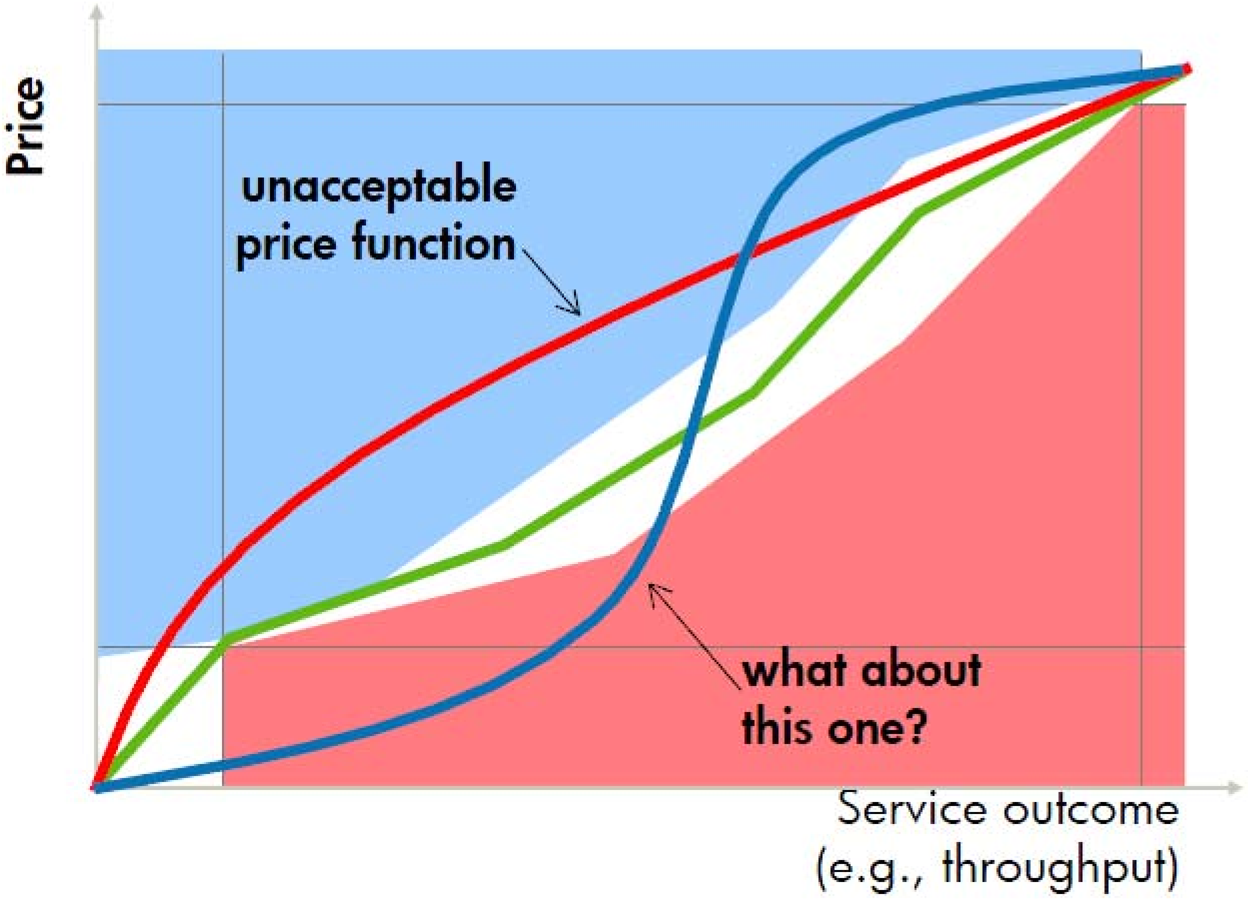
**John Wilkes, HP Labs: Utility functions, prices, and negotiation, HP Tech Report and Slides, 2008.**

# Room for Negotiation



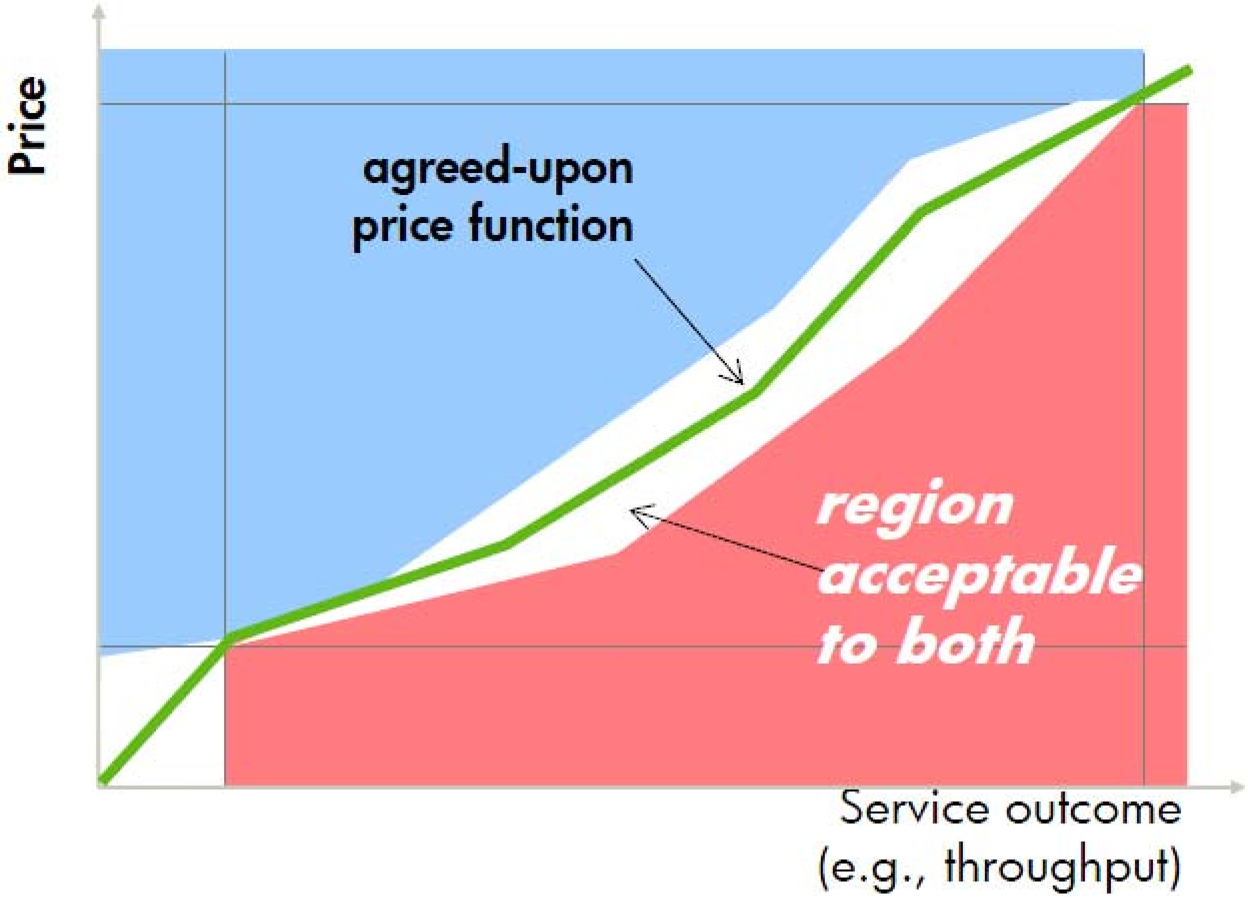
**John Wilkes, HP Labs: Utility functions, prices, and negotiation, HP Tech Report and Slides, 2008.**

# Room for Negotiation



**John Wilkes, HP Labs: Utility functions, prices, and negotiation, HP Tech Report and Slides, 2008.**

# Negotiated Price Function



**John Wilkes, HP Labs: Utility functions, prices, and negotiation, HP Tech Report and Slides, 2008.**