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

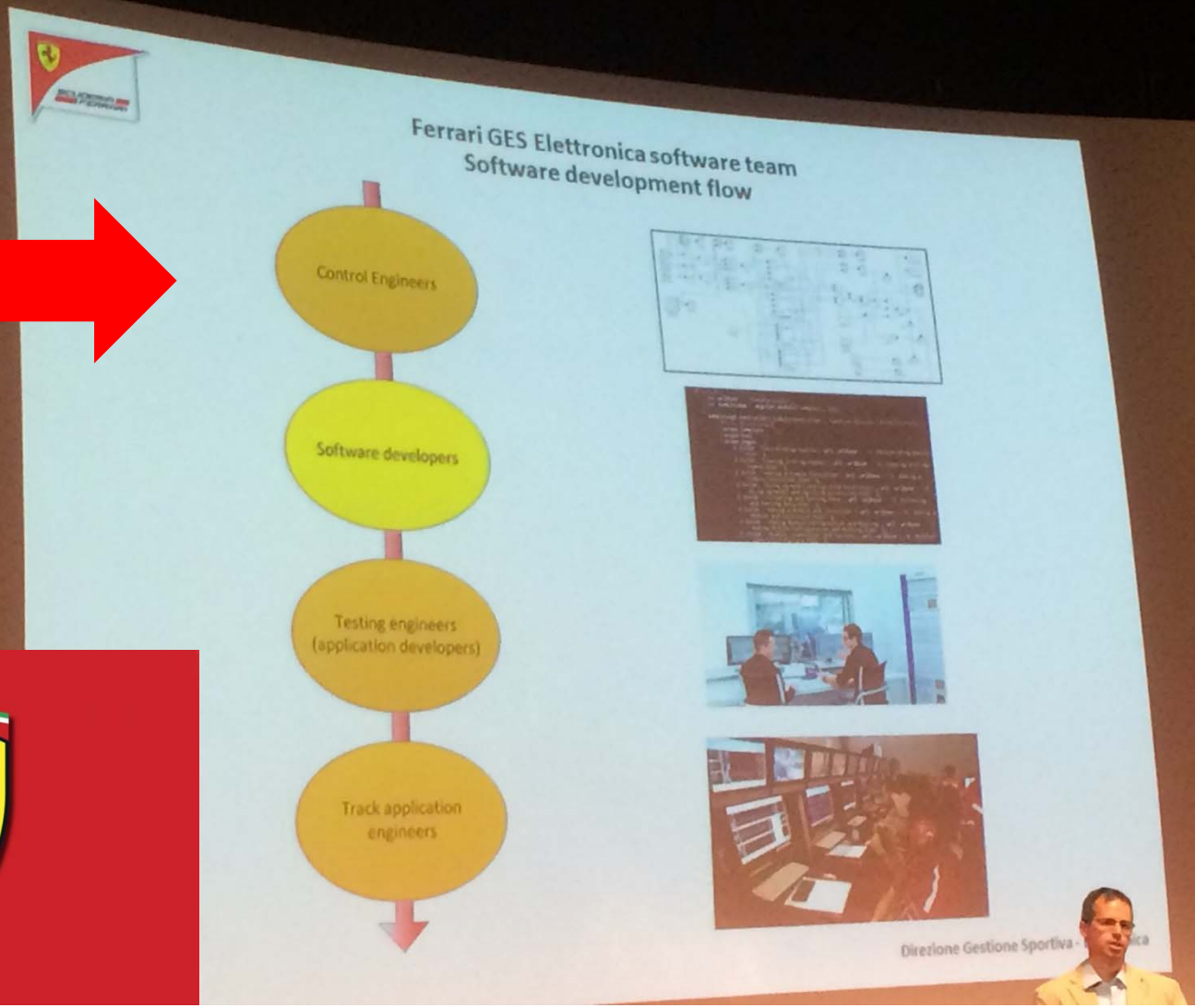
Announcements

- Friday, June 19
  - A2 due
- A3
  - Posted and due July 10
- Grad project
  - Handed out June 22
  - Due July 25
- Marks posted
  - A1 — Avg: 86.8; Med: 88
  - M1 — Avg: 78.9; Med: 86

Ferrari CPS Keynote

Claudio Silenzi

2015.icse-conferences.org/program/keynotes



2015 IEEE Computer Society Board of Governors

Need your help

Catchy acronym for grant proposal

Consortium for
Cyber Physical Systems Research

If yours is selected $25 for your OneCard ☺

Deadline 2 pm today: hausimuller@gmail.com

Autonomic Computing Vision

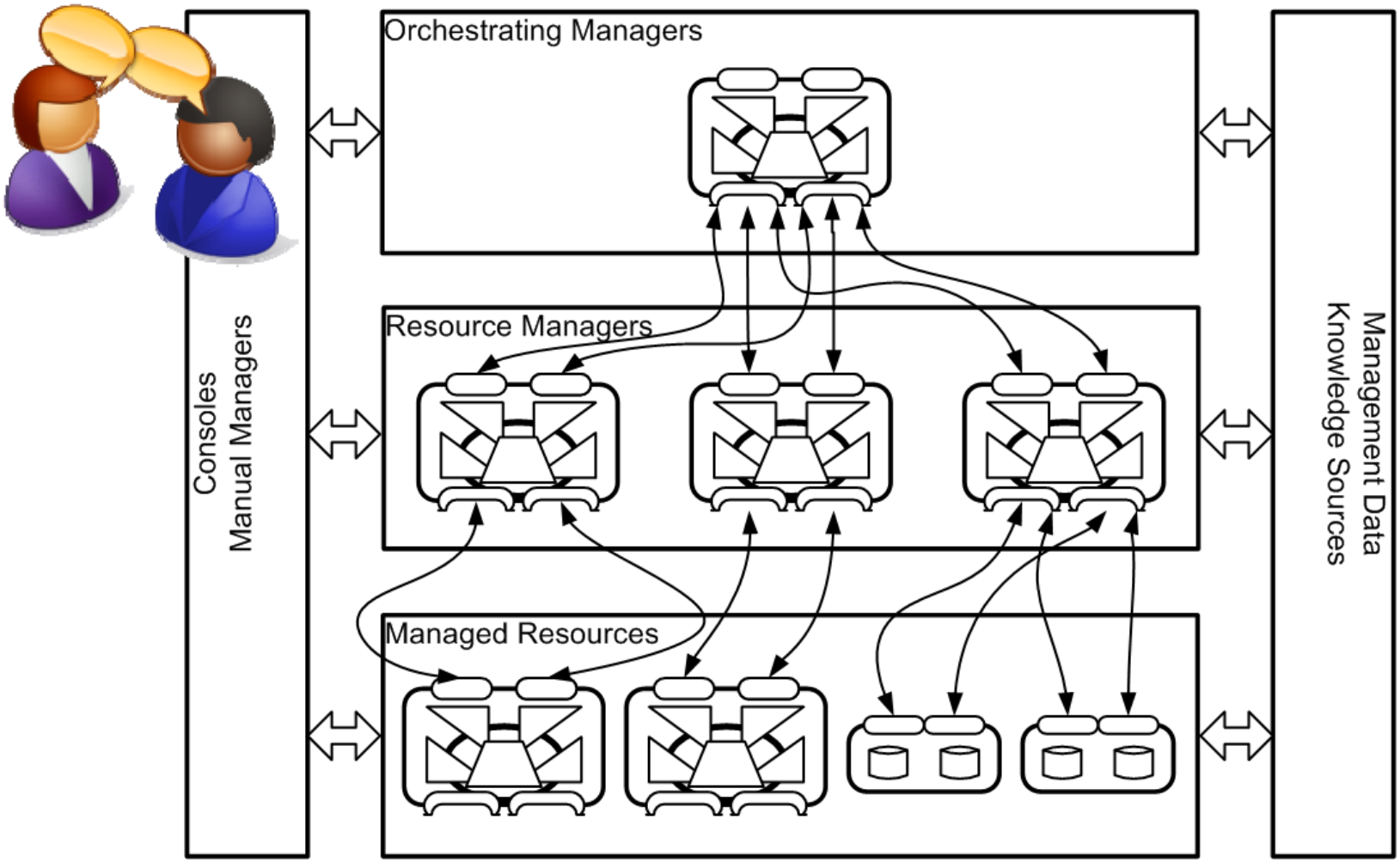
Autonomic Computing is really about making systems self-managing …

—Paul Horn, IBM Research, 2001

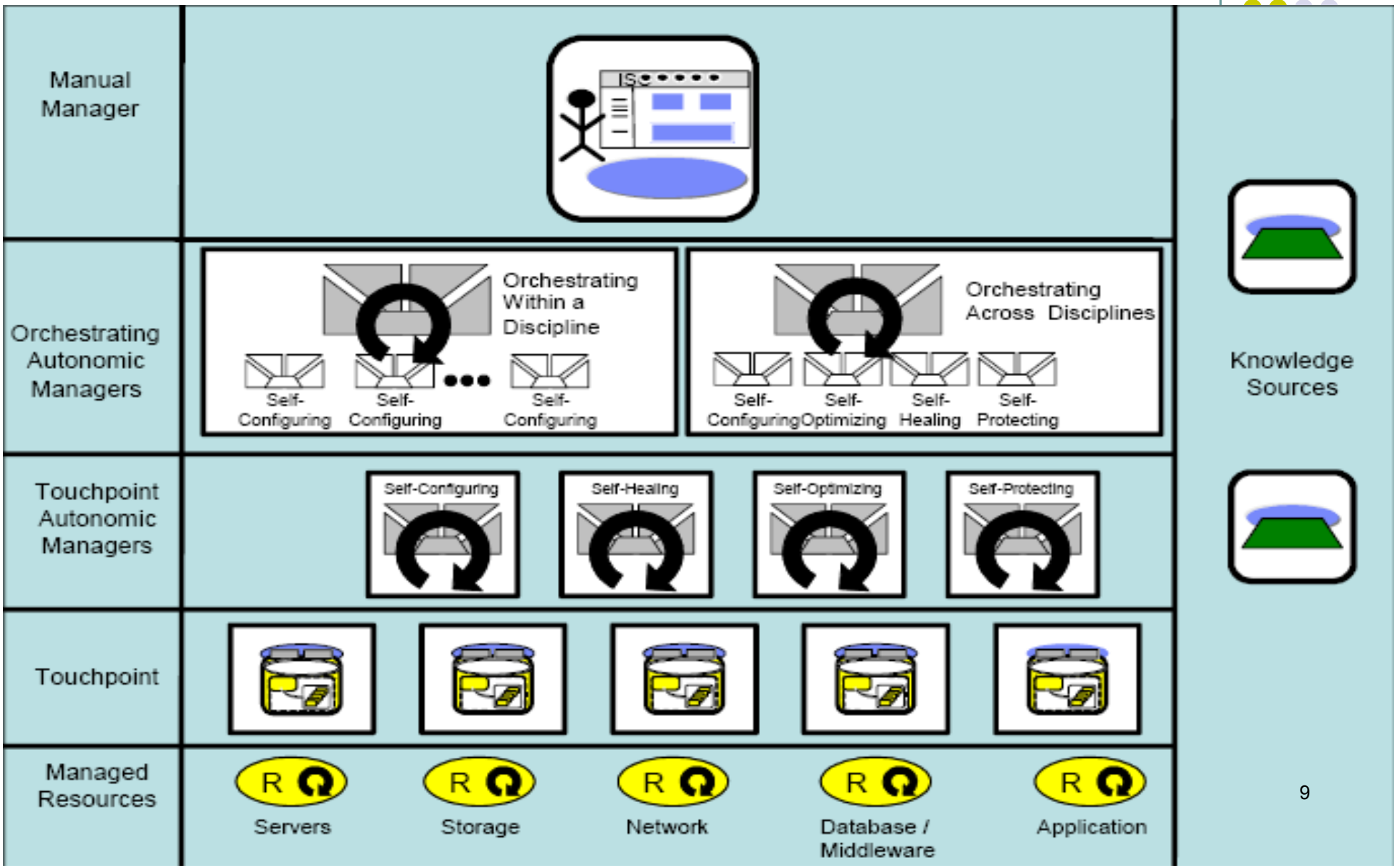
Reading Assignment

- Kephart, J.O., Chess, D.M.: The Vision of Autonomic Computing. IEEE Computer 36(1):41-50 (2003)
  ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1160055
- IBM: An Architectural Blueprint for Autonomic Computing, 4th Ed. (2006)
  citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.150.1011&rep=rep1&type=pdf

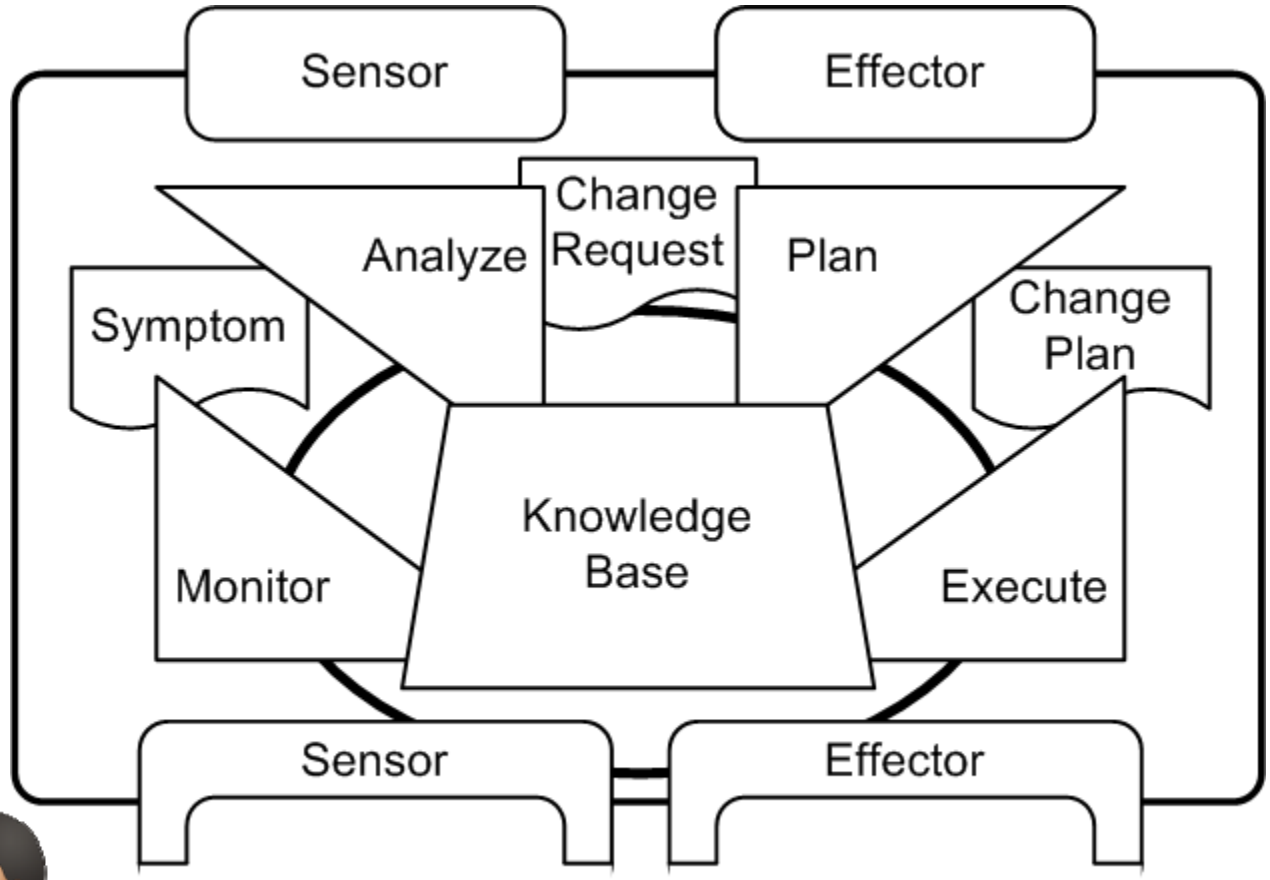
ACRA
AC Reference Architecture



ACRA
AC Reference Architecture



Autonomic Manager



Autonomic Managers Implement Self-* MAPE-K Loops

Increased Responsiveness

Adapt to dynamically changing environments

Operational Efficiency

Tune resources and balance workloads to maximize use of IT resources

Business Resiliency

Discover, diagnose, and act to prevent disruptions

Secure Information and Resources

Anticipate, detect, identify, and protect against attacks

*Self – **

Useful Papers under Resources Course Web Site

- Ganek, A.G., Corbi, T.A.: The Dawning of the Autonomic Computing Era. IBM Systems Journal 42(1):5-18 (2003)
- Kephart, J.O., Chess, D.M.: The Vision of Autonomic Computing. IEEE Computer 36(1):41-50 (2003)
- Kluth, A.: Information Technology: Make It Simple. The Economist (2004)

- Huebscher, M.C., McCann, J.A.: A Survey of Autonomic Computing—Degrees, Models, and Applications. ACM Computing Surveys, 40 (3):7:1-28 (2008)
- Müller, H.A., Kienle, H.M., Stege, U.: Autonomic Computing: Now You See It, Now You Don't—Design and Evolution of Autonomic Software Systems. In: De Lucia, A.; Ferrucci, F. (eds.): Software Engineering International Summer School Lectures: University of Salerno. LNCS, Springer-Verlag, Heidelberg, pp. 32–54 (2009)

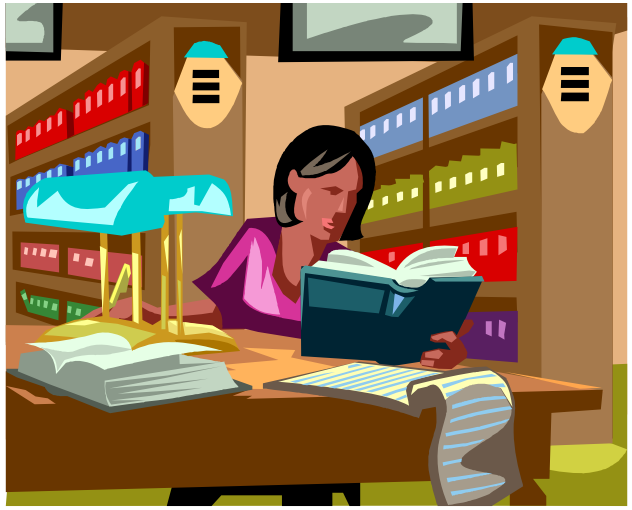
- Dobson, S., Denazis, S., Fernandez, A., Gaiti, D., Gelenbe, E., Massacci, F., Nixon, P., Saffre, F., Schmidt, N., Zambonelli, F.: A Survey of Autonomic Communications. ACM Transactions on Autonomous and Adaptive Systems (TAAS) 1(2):223-259 (2006)

Useful Papers under Resources Course Web Site

- Diao, Y., Hellerstein, J.L., Parekh, S., Griffith, R., Kaiser, G.E., Phung, D.: A Control Theory Foundation for Self-Managing Computing Systems. IEEE Journal on Selected Areas in Communications 23(12):2213-2222 (2005)
- Müller, H.A., Pezzè, M., Shaw, M.: Visibility of Control in Adaptive System. In: 2nd ACM/IEEE International ICSE Workshop on Ultra-Large-Scale Software-Intensive Systems (ULSSIS 2008), pp. 23-26, ACM, New York, NY, USA (2008)
- Dawson, R., Desmarais, R., Kienle, H.M., Müller, H.A.: Monitoring in Adaptive Systems Using Reflection. In: 3rd ACM/IEEE International ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2008), pp. 81-88, ACM, New York, NY, USA (2008)
- 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)
- IBM: Symptoms Reference specification Version 2.0 (2006)

Useful Papers under Resources Course Web Site

- Study these papers
- Immerse yourself in the autonomic computing literature and technology
- Huge asset for your job application and future job

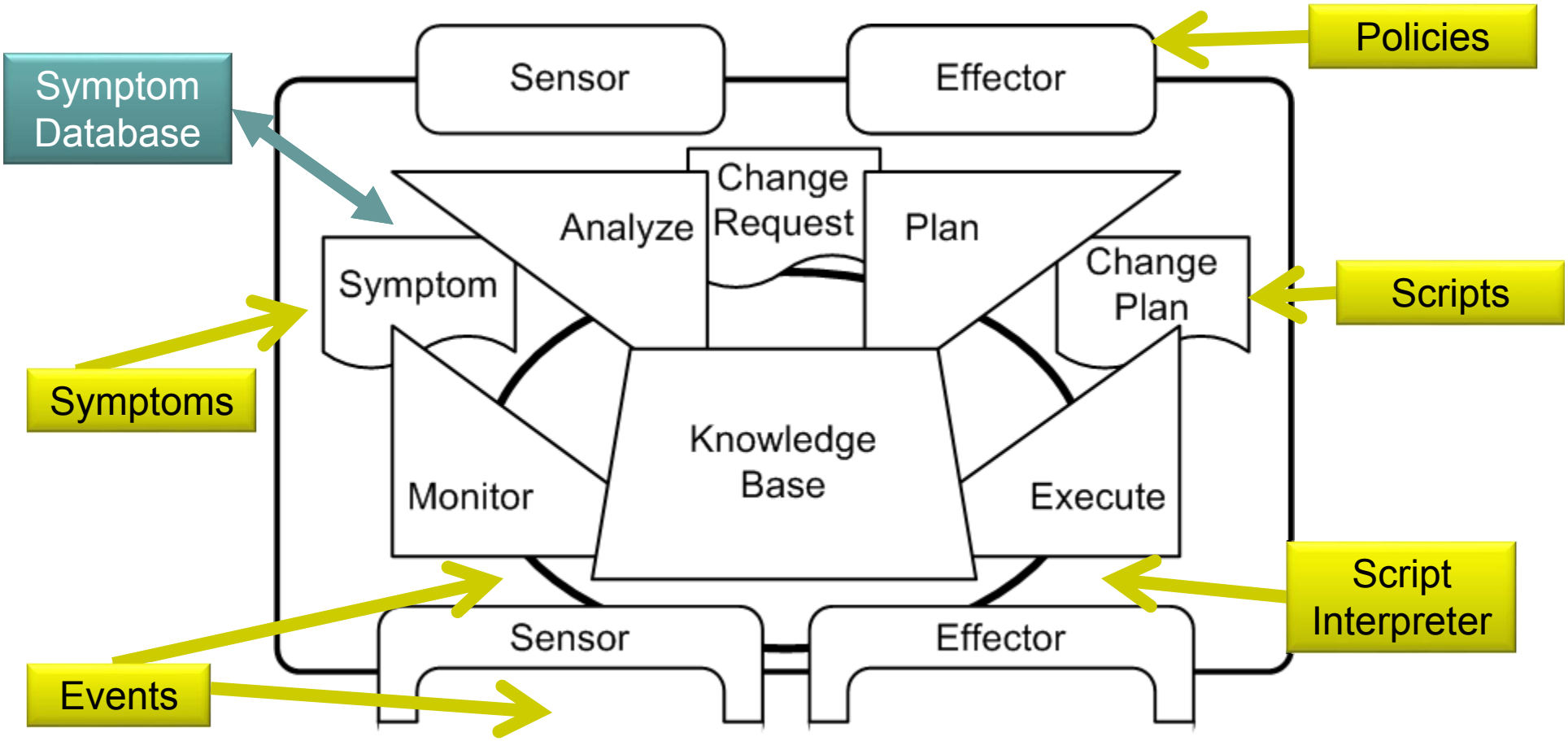


Implementing Autonomic Elements

The devil lies in the details ...

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

MAPE-K Loop Standards & Interfaces



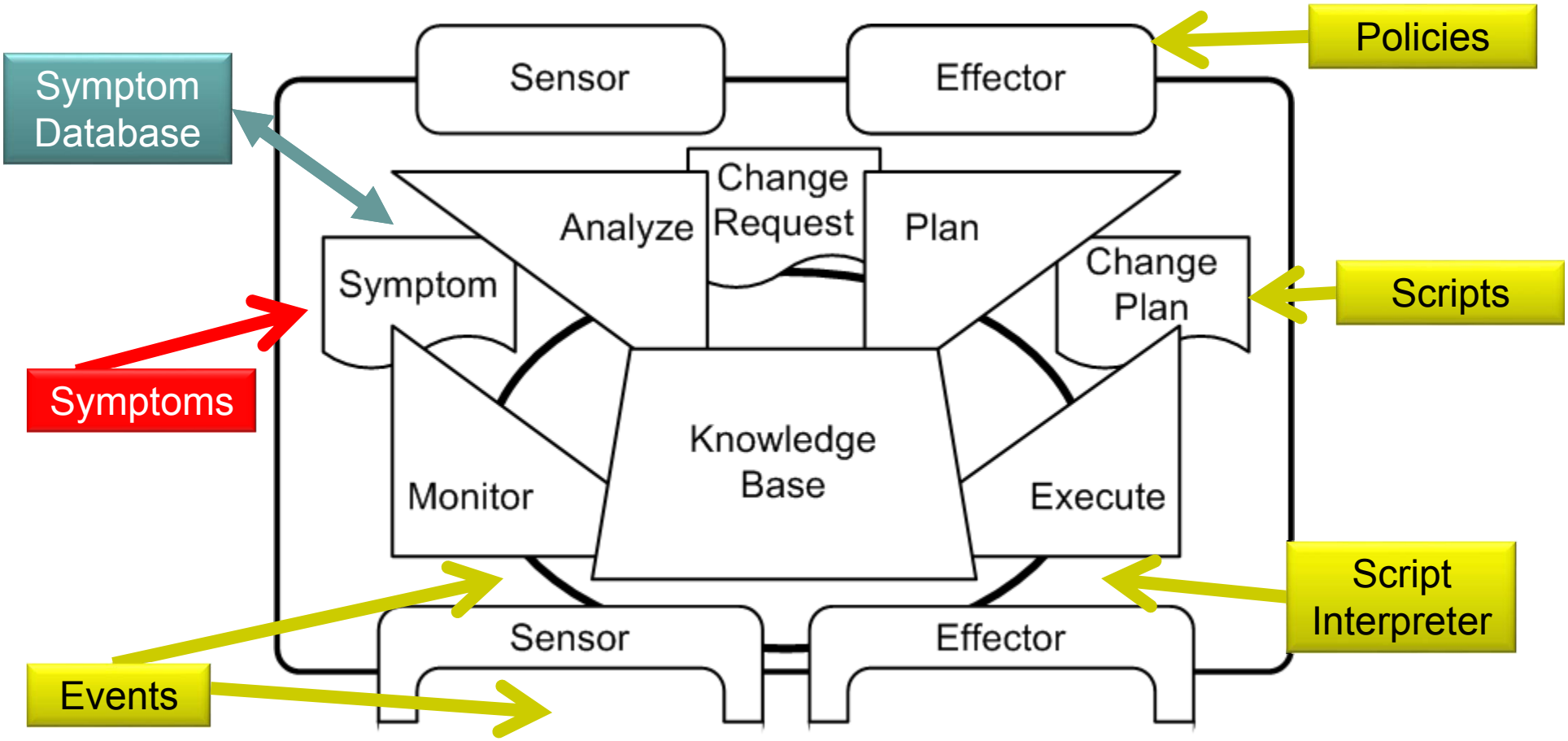
Information Interchange in the ACRA Architecture

- What information is passed between the components of an autonomic architecture adhering to the ACRA reference architecture?
- Information is exchanged in the form of events and knowledge in the knowledge bases
- Ideally the exchanged information is standardized
  - Formats
  - Schemas
- Information is exchanged between the manager and the managed element
  - Events
  - Set and get operations
- Policies are injected into autonomic elements through the effectors on top of the manager
- Information is passed around the MAPE-K loop

Events

- An event is an asynchronous state transition in the managed element
- Events are generated by managed elements and are processed by autonomic managers
- Event processing is a discipline that aims to define and develop
  - Event abstractions
  - Event architectures
  - Event systems
  - Event languages
  - Event patterns
  - Event models
  - Event processing standards
  - Event exchange standards

MAPE-K Loop Standards & Interfaces



Symptom Definition

- Symptom definitions
  - Is an artefact used to recognize a symptom occurrence
  - Must be compatible with their respective correlation engines
- A symptom definition can be anything
  - XPATH expression
  - Regular expression
  - Decision tree
  - Dependency graph
  - Prolog predicate
  - ACT pattern
  - TEC rule
  - Neural network

Symptom Examples

Symptom name	Symptom description	Symptom definition	Symptom recommendations
Authentication failure	Attempt to access resources associated with this symptom was made, but there was an authentication failure	Collection pattern: event(wrong_password) n=3 timeout=24h	Log for auditing purposes
Authorization failure	Unauthorized attempt to access resources associated with this symptom was made, and access was denied	Filter pattern: event(access_denied)	Log for auditing purposes
Prevention deployment failure	Failure occurred while deploying security prevention resources (virus update table, security patch, and so on)	Filter pattern: event(security_install_failed)	Analyze security prevention failure *and* alert security administrator

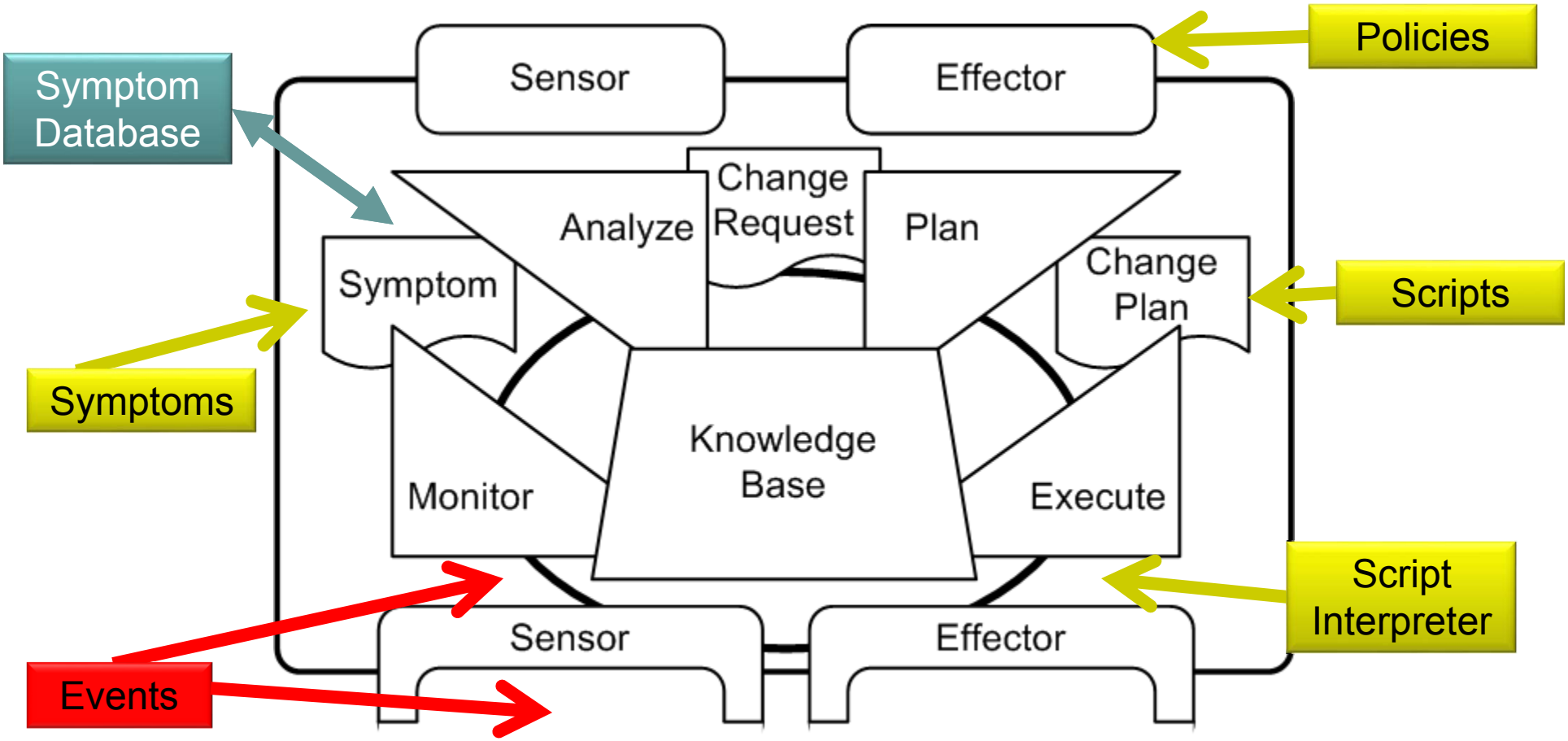
M. Perazolo, IBM: Symptoms deep dive
Part 2: Cool things you can do with symptoms, Dec 2005

Symptom Examples

Symptom name	Symptom description	Symptom definition	Symptom recommendations
Configuration unavailable	Some configuration information for the resources associated with this symptom was not found	Filter pattern: event(configuration_not_found)	Alert administrator and flag service provided by resource as "marginal"
Configuration invalid	Configuration information for the resources associated with this symptom was processed and determined to be invalid	Sequence pattern: event(configuration_found) event(configuration_invalid)	Alert administrator and flag service provided by resource as "marginal"
Dependency unavailable	One or more dependencies (resources) are non-existent and needed by other resources	Sequence pattern: event(dependency_request, resource) event(inventory, resource not within [inventory_list])	Install missing resource
Dependency mismatch	Release level of one or more resources associated with this symptom are not what was expected	Filter pattern: event(wrong_release)	Update resource to required release

M. Perazolo, IBM: Symptoms deep dive
Part 2: Cool things you can do with symptoms, Dec 2005

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)

Linking AM and ME using Standardized Web Services

- MOWS
  - OASIS: Web Services Distributed Management: Management of Web Services (WSDM-MOWS) 1.1 OASIS Standard (2006)
- WUWS
  - OASIS: Web Services Distributed Management: Management Using Web Services (WSDM-MUWS) 1.1 OASIS Standard (2006)
- AC and Standardized WS
  - Kreger, H., Studwell, T.: Autonomic Computing and Web Services Distributed Management (2005)
- All leading system management suppliers participated in this committee

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

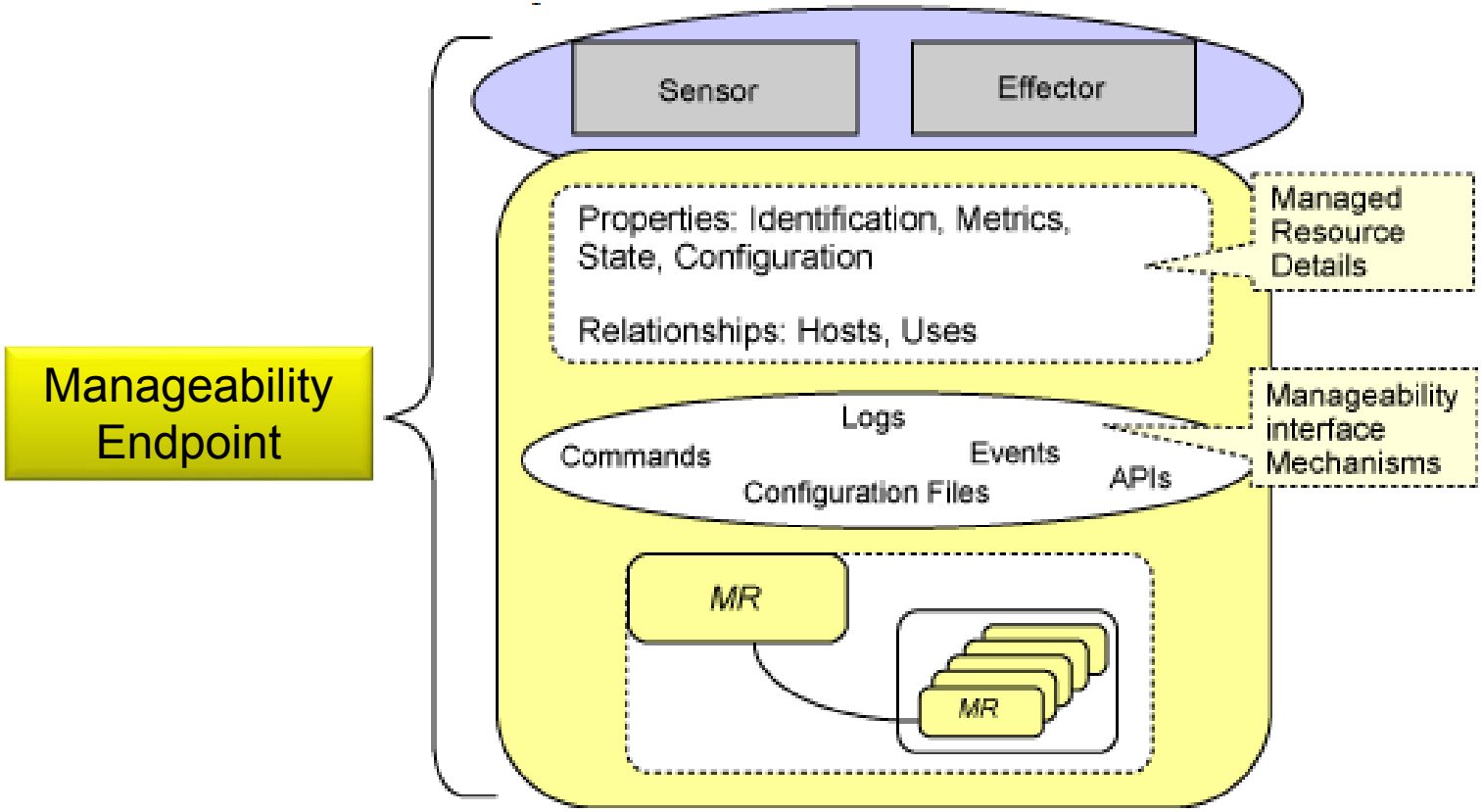
Manageability Endpoint and Interface

- Manageability Endpoint (ME)
  - A manageability endpoint is the component in a system that exposes the state and management operations for a resource in the system. An autonomic manager communicates with a manageability endpoint through the manageability interface.
- Manageability Interface (MI)
  - A manageability endpoint is the implementation of the manageability interface for a specific manageable resource or a set of related manageable resources.

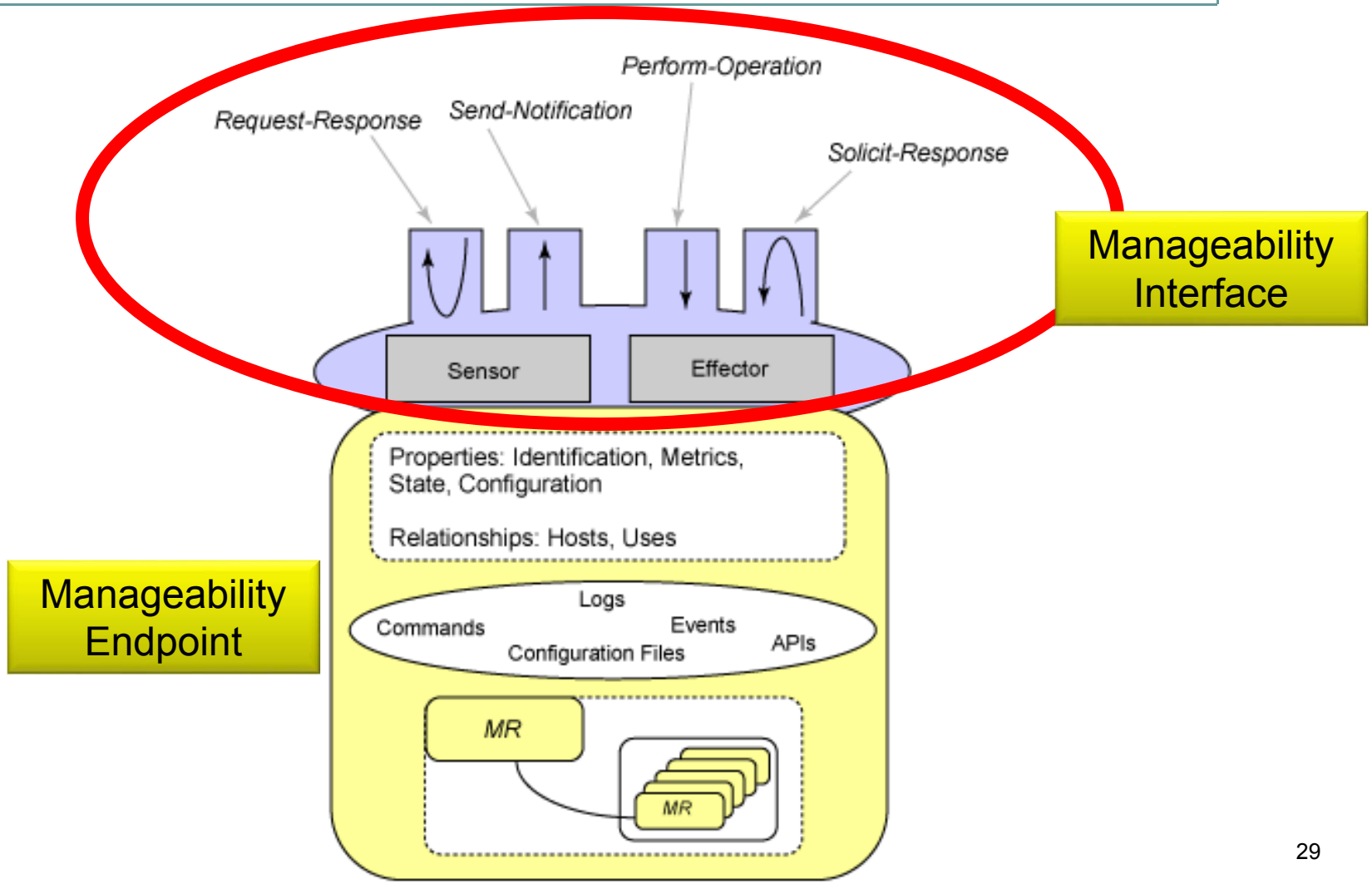
- Standards-based management interfaces and data formats

IBM: An Architecture Blueprint for Autonomic Computing, 4th Ed. 2006

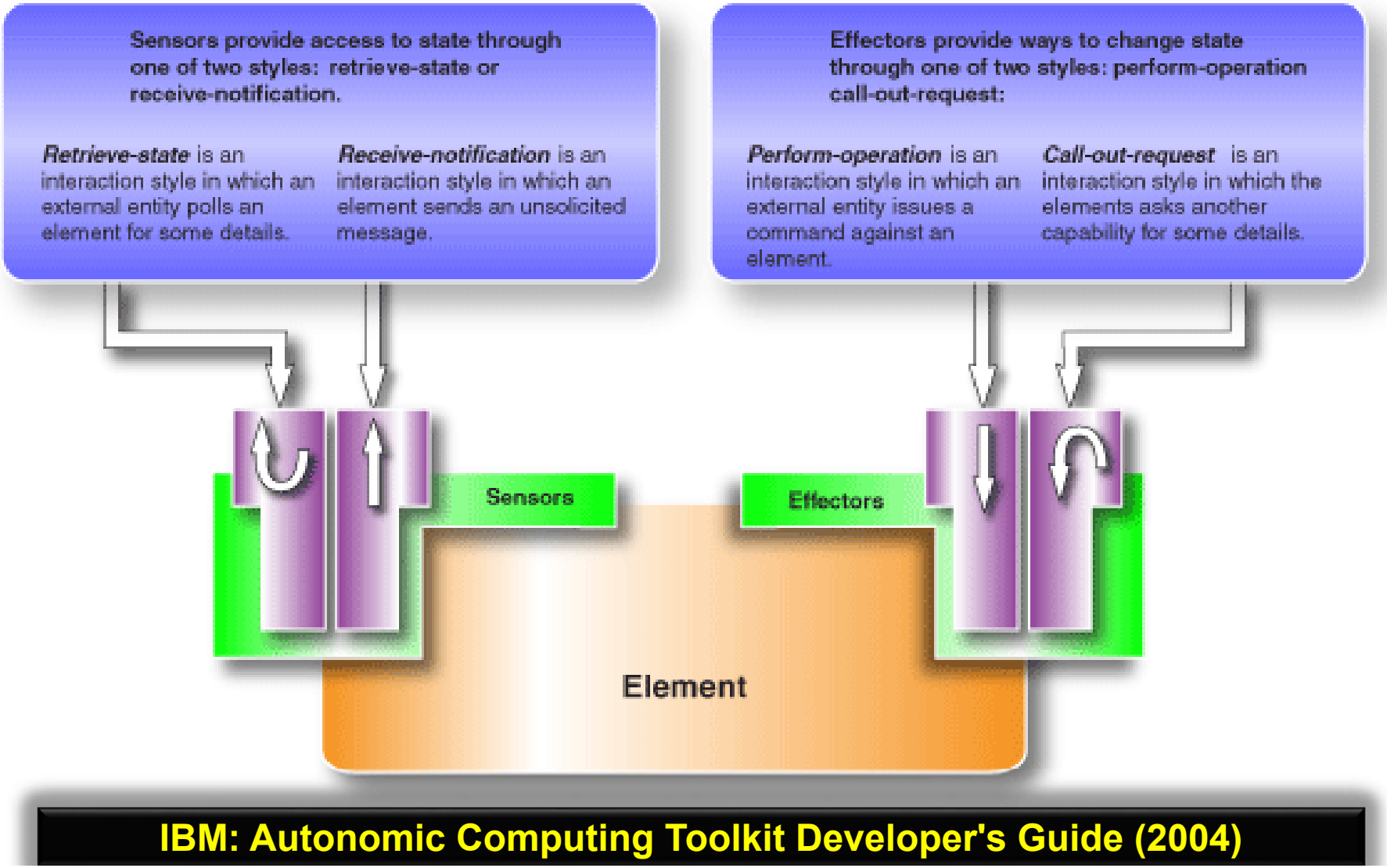
Manageability Endpoint



Manageability Interface



Manageability Interface Interaction Styles



IBM: Autonomic Computing Toolkit Developer's Guide (2004)

Manageability Interface Interaction Styles

Sensor retrieve-state

- Used by an AM to query state information from an ME
- The AM asks for information and the ME synchronously returns it

Sensor receive-notification

- A ME uses this style to asynchronously send event information to an AM

Effector perform-operation

- Used by an AM to issue a command to an ME
- Used to change states or properties in the endpoint

Effector call-out-request

- Used by a ME to consult with an external entity before taking certain actions—to check what changes are allowed prior to changing values
- Used to gather information from an AM before making a change

Manageability Endpoint Infrastructure

- The IBM Manageability Endpoint Builder
  - Includes tools and a run-time environment for building endpoints that allow products to expose manageability interfaces using the WSDM standard
  - Is accessible from a standard Eclipse environment and from the IBM Rational Application Developer product
  - With this interface, any WSDM-compliant tool or autonomic manager can view the status of the resource and make calls to modify the resource's state
- The IBM Manageability Endpoint Simulator
  - Assists in the development of autonomic managers by emulating a WSDM-compliant managed resource
  - A major hurdle in building autonomic managers is that developers need resources (and endpoints) with which to test
  - The Manageability Endpoint Builder addresses this problem.

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

WSDM Endpoints

- Web Services Addressing (WS-Addressing)
  - W3C Standard: http://www.w3.org/Submission/ws-addressing/
  - <wsa:EndpointReference>
    <wsa:Address>*xs:anyURI*</wsa:Address>
    <wsa:ReferenceProperties>... </wsa:ReferenceProperties> ?
    <wsa:ReferenceParameters>... </wsa:ReferenceParameters> ?
    <wsa:PortType>*xs:QName*</wsa:PortType> ?
    <wsa:ServiceName PortName="*xs:NCName*"?>*xs:QName*</wsa:ServiceName> ?
    <wsp:Policy> ... </wsp:Policy>*
    </wsa:EndpointReference>
- Web Services Distributed Management (WSDM)
  - The open standard WSDM is supported by two open source projects: a reference implementation in the Apache Muse project and tooling in the Eclipse TPTP (Test & Performance Tools Platform) project
  - Interactively test your WSDM endpoints in real-time using the Eclipse TPTP tooling
  - http://www.ibm.com/developerworks/library/acmanexp1/#connect

Overview of WSDM (Pronounced Wisdom) Web Services Distributed Management

- MUWS defines how to represent and access the manageability interfaces of resources as Web services
  - Standard manageable resource definitions create an integration layer between managers and the different management protocols used to instrument resources
  - They are the foundation of enabling the use of Web services to build management applications and allowing many managers with one set of instrumentation to manage resources
- MOWS defines how to manage Web services as resources and how to describe and access that manageability using MUWS
  - Provides mechanisms and methodologies that enable manageable Web services applications to interoperate across enterprise and organizational boundaries
  - MOWS allows integration of management with Web services-based business applications and processes.

WSDM Architectural Objectives

- Architectural foundations
  - Web services
  - Service-oriented architecture (SOA)
  - Underlying standards: XML, SOAP, WSDL
- Architectural objectives
  - Resource oriented
  - Implementation isolation
  - Composeability of services
  - Model agnostic
  - Enabling inspection

WSDM Architectural Objectives

- Resource oriented
  - Historically, managers have accessed resources through management agents running on the resource.
  - By describing and offering resource access interfaces for resources directly rather than through intermediaries, WSDM makes resources Web services which can now participate directly in a service oriented architecture and business processes.
- Implementation isolation
  - Isolates manageable resources access from their manageable resource implementations.

WSDM Architectural Objectives

- Composeability of services
  - To scale, the specification takes advantage of the composeability of services afforded by Web services architectures.
  - Stack of resource and web service endpoints
- Uniform manageability model
  - WSDM describes HOW to access management data pertaining to managed resources by means of a Web service protocol.
  - Manageability capabilities
- Enabling inspection
  - enables inspection (or discovery) of resource interfaces (properties, operations and events) at design time and run time

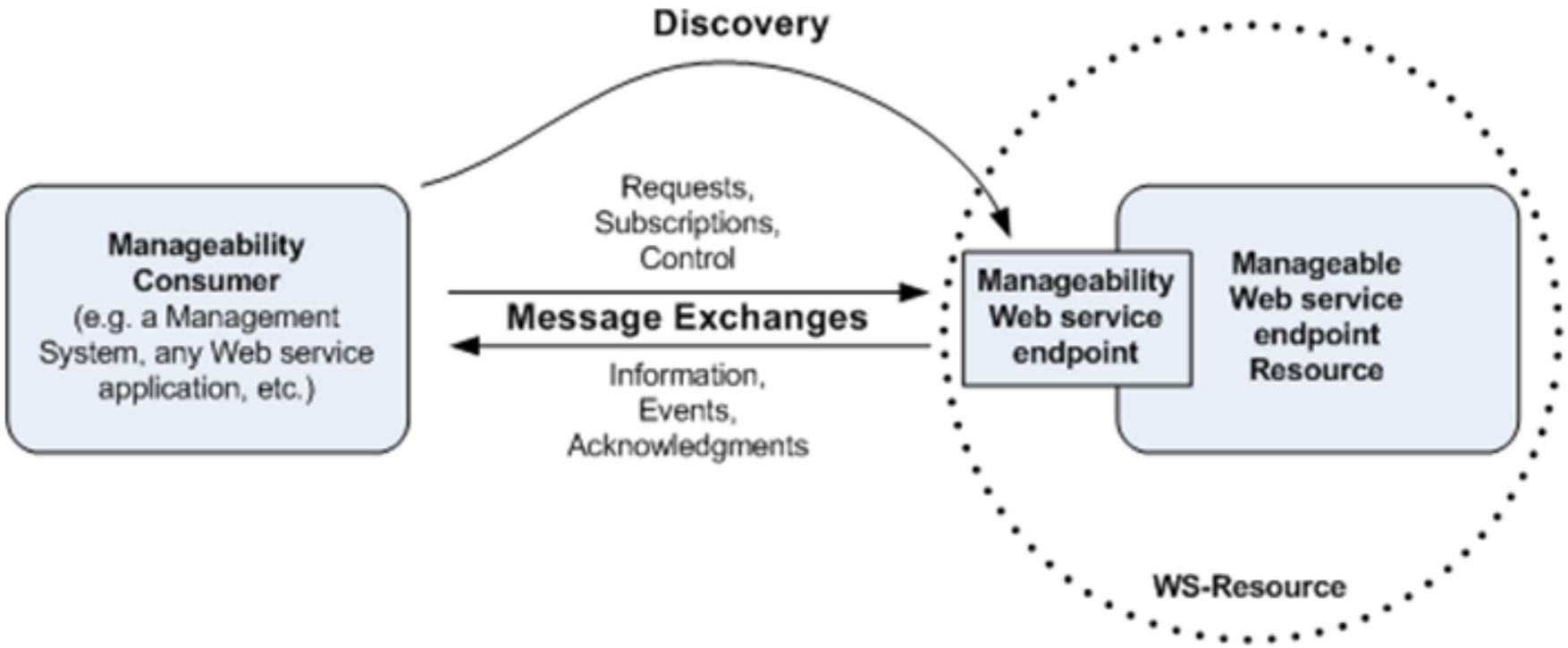
MOWS: Management Of Web Services Web Services Endpoints

- Web services are an integral part of the IT landscape
- Autonomic managers are often used to mange web services
- Web services can be used by autonomic managers to communicate with managed elements
- To manage a web services, one needs to manage the web services endpoints
- The WSDM-MOWS specification addresses the management of the web services endpoints using web services protocols

MUWS

- MUWS defines
  - how the ability to manage, or
  - how the *manageability of*, an arbitrary *resource* can be made accessible via *Web services*
- In order to achieve this goal, MUWS is based on a number of Web services specifications, mainly for messaging, description, discovery, accessing properties, and notifications

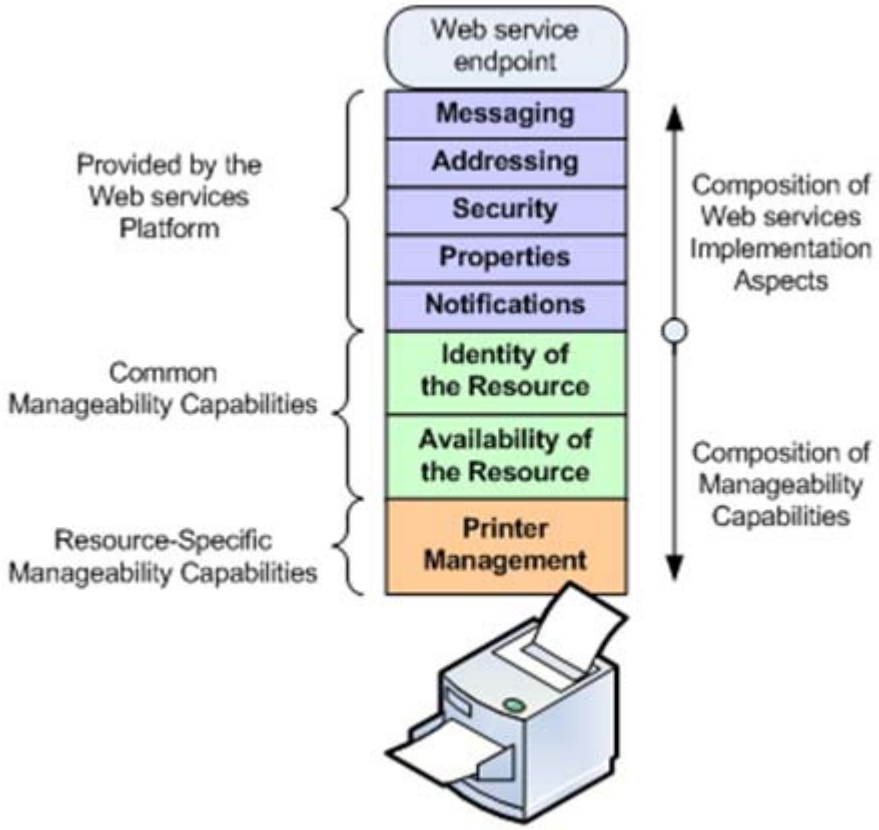
MOWS: Management Of Web Services Web Services Endpoints



OASIS: Web Services Distributed Management: Management of Web Services (WSDM-MOWS) 1.1 OASIS Standard (2006)

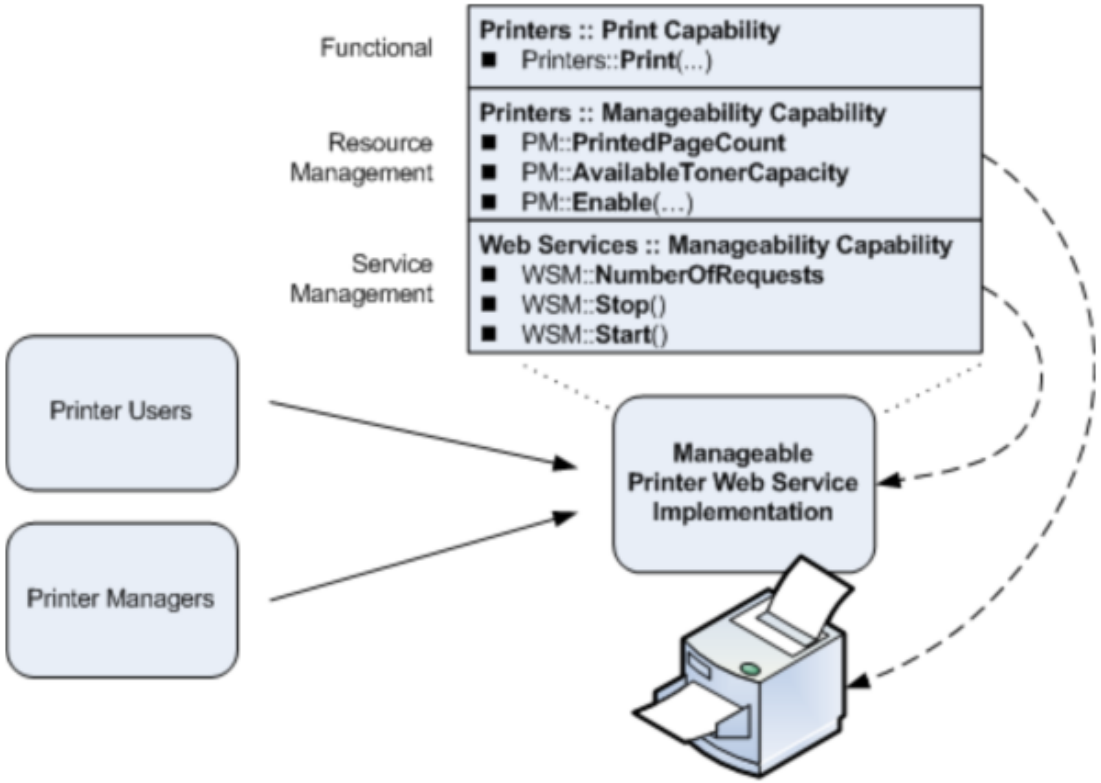
Composition of Resource Endpoint and Web Service Endpoint

- The composition as implemented by a manageable resource is then accessible via a web service endpoint



Resources Exposed as Web Services

- WSDM allows a resource and its services to be manageable in a standard manner
- A resource may support both manageability and functional capabilities
- A printer can print and may indicate if it is on-line (functional) and may be able to notify when the toner is running out (manageability)
- Select an on-line printer with sufficient amount of toner in order to print an urgent report for executives

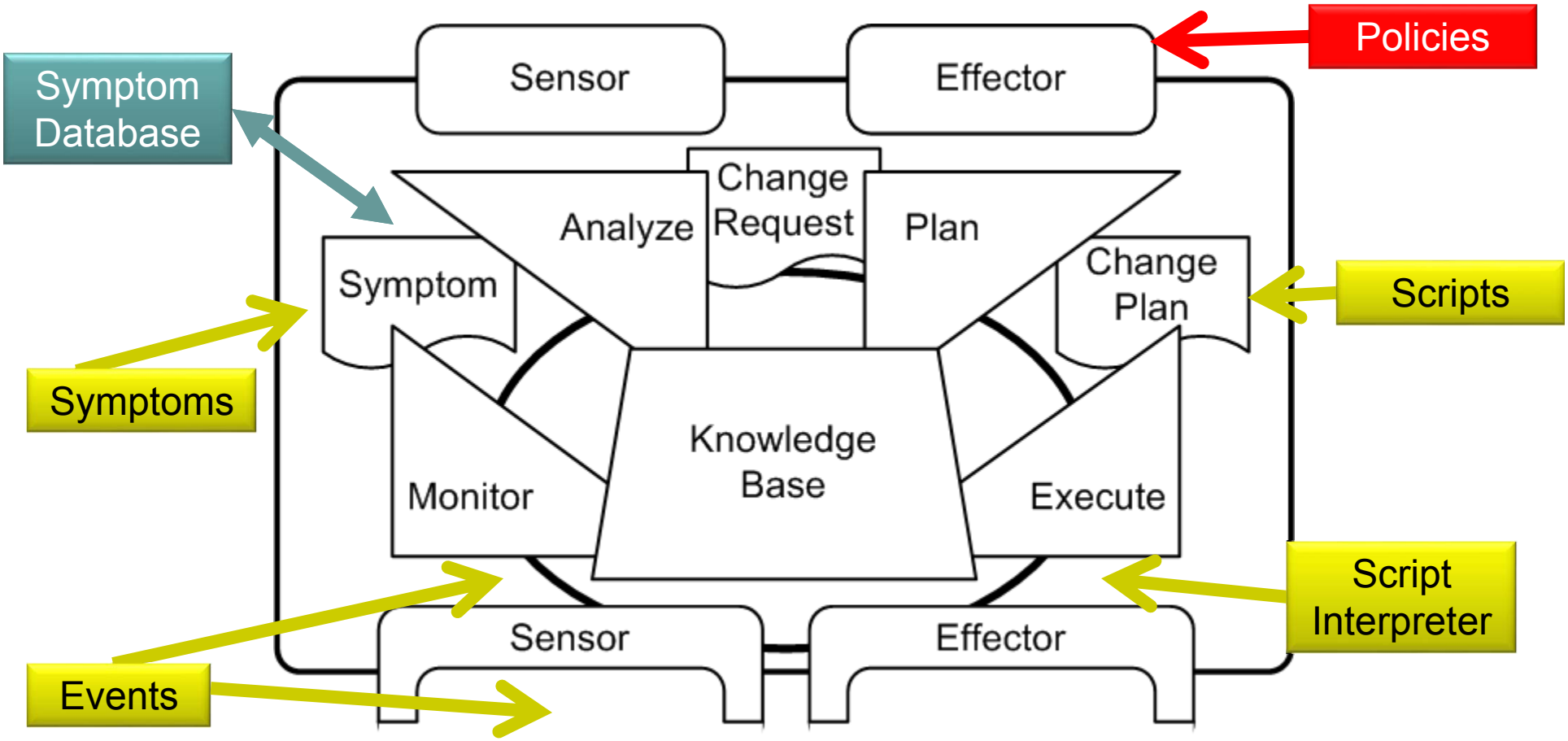


OASIS: Web Services Distributed Management: Management of Web Services (WSDM-MOWS) 1.1 OASIS Standard (2006)

Web Service Manageability Capabilities

- Common manageability capabilities
- Web service endpoint manageability capabilities
- Discover web service endpoints
- Discover capabilities
- Use capabilities
- The road to autonomic computing using service-oriented architecture (SOA)

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

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