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

Announcements

- Week of June 18, 19, 21
  - Instructor: Ron Desmarais
  - PID controllers
  - PID controller is a generic control loop feedback controller widely used in industrial control systems
- Müller in Montréal
  - Elected Fellow of CAE
- Thu, June 20
  - Assignment 2 due
- Fri, June 28
  - Midterm in class

THE CANADIAN ACADEMY OF ENGINEERING
Leadership in Engineering Advice for Canada

L'ACADÉMIE CANADIENNE DU GÉNIE
Chef de file en matière d'expertise-conseil en génie pour le Canada

The Canadian Academy of Engineering (CAE) comprises many of the country's most accomplished engineers, who have expressed their dedication to the application of science and engineering principles in the interests of the country and its enterprises. The Academy is an independent, self-governing and non-profit organization established in 1987 to serve the nation in matters of engineering concern.

The Academy is an active member of the International Council of Academies of Engineering and Technological Sciences (CAETS), which involves 26 leading countries.

Members of the Academy are nominated and elected by their peers to honorary Fellowships, in view of their distinguished achievements and career-long service to the engineering profession. Members work closely with the other national engineering associations in Canada, and with the other Canadian academies that comprise the Council of Canadian Academies.

Council of Canadian Academies
Conseil des académies canadiennes

About the Council and Academies

The Council of Canadian Academies (the Council) The Council of Canadian Academies is an independent, not-for-profit corporation that began operation in 2005. The Council supports evidence-based, expert assessments to inform public policy development in Canada. Assessments are conducted by independent, multidisciplinary panels of experts from across Canada and abroad. The Council's blue-ribbon panels serve free of charge and many are Fellows of the Council's Member Academies. The Council defines science broadly to include the natural, social and health sciences, engineering and the humanities. The Council's vision is to be Canada's trusted voice for science in the public interest.

Canadian Academy of Health Sciences
Académie canadienne des sciences de la santé

RSC
The Royal Society of Canada
The Academies of Arts, Humanities and Sciences of Canada

SRC
La Société royale du Canada
Les Académies des arts, des lettres et des sciences du Canada

THE CANADIAN ACADEMY OF ENGINEERING
Leadership in Engineering Advice for Canada

L'ACADÉMIE CANADIENNE DU GÉNIE
Chef de file en matière d'expertise-conseil en génie pour le Canada

The Royal Society of Canada (RSC) is the senior national body of distinguished Canadian scholars, artists and scientists. The primary objective of the RSC is to promote learning and research in the arts and sciences. The RSC consists of nearly 2,000 Fellows – men and women who are selected by their peers for outstanding contributions to the natural and social sciences, the arts and the humanities. The RSC exists to recognize academic excellence, advise governments and organizations, and promote Canadian culture.

The Canadian Academy of Engineering (CAE) is the national institution through which Canada's most distinguished and experienced engineers provide strategic advice on matters of critical importance to Canada. The Academy is an independent, self-governing, and non-profit organization established in 1987. Members of the Academy are nominated and elected by their peers to honorary fellowships, in recognition of their distinguished achievements and career-long service to the engineering profession. Fellows of the academy are committed to ensuring that Canada's engineering expertise is applied to the benefit of all Canadians.

The Canadian Academy of Health Sciences (CAHS) recognizes individuals of great achievement in the academic health sciences in Canada. Founded in 2004, the CAHS has approximately 400 Fellows and appoints new Fellows on an annual basis. The organization is managed by a voluntary Board of Directors and a Board Executive. The main function of CAHS is to provide timely, informed, and unbiased assessments of urgent issues affecting the health of Canadians. The Academy also monitors global health-related events to enhance Canada's state of readiness for the future, and provides a Canadian voice for health sciences internationally. CAHS provides a collective, authoritative, multi-disciplinary voice on behalf of the health sciences community.

A Statement of Common Understanding

Among the Council of Canadian Academies, the Royal Society of Canada, the Canadian Academy of Engineering & the Canadian Academy of Health Sciences

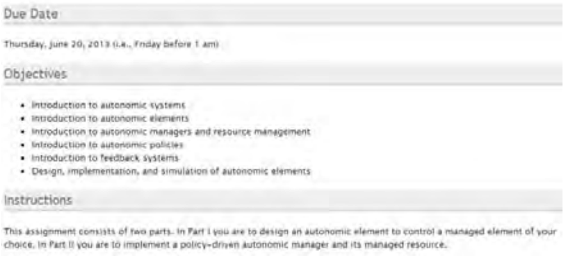
In 2011 the presidents of the Council of Canadian Academies, the Royal Society of Canada, the Canadian Academy of Engineering and the Canadian Academy of Health Sciences worked cooperatively to develop a joint *Statement of Common Understanding* to guide their future collaboration. The goal of the agreement is to bring together intellectual resources in synergistic ways to generate capacity for credible, evidence-based, and independent scientific advice in support of the development of sound public policy in Canada.

The development of a strong collaborative partnership among the four organizations will:

- Promote trust and understanding of collective and individual organizational goals;
- Help leverage each organization's strengths and identify complementarities in expertise, capabilities, knowledge and talent;
- Create synergistic use of resources in an efficient and cost-effective manner (whether it is funding, expert volunteers or operational capacity); and
- Provide opportunity for innovative thinking in the provision of expert scientific advice.

All parties recognize that success will require strong mutual reliance among the four organizations and a long-term commitment.

Assignment 2 Instructions



Due Date

Thursday, June 20, 2013 (i.e., Friday before 1 am)

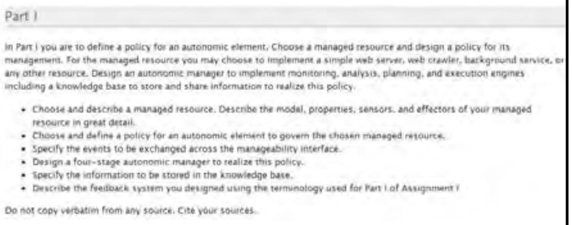
Objectives

- Introduction to autonomic systems
- Introduction to autonomic elements
- Introduction to autonomic managers and resource management
- Introduction to autonomic policies
- Introduction to feedback systems
- Design, implementation, and simulation of autonomic elements

Instructions

This assignment consists of two parts. In Part I you are to design an autonomic element to control a managed element of your choice. In Part II you are to implement a policy-driven autonomic manager and its managed resource.

Assignment 2 Part I



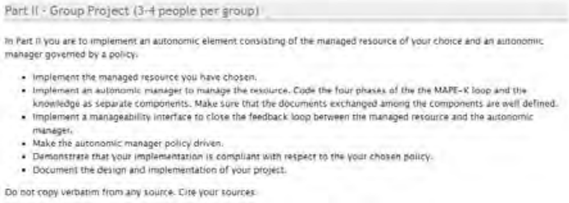
Part I

In Part I you are to define a policy for an autonomic element. Choose a managed resource and design a policy for its management. For the managed resource you may choose to implement a simple web server, web crawler, background service, or any other resource. 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 you designed using the terminology used for Part I of Assignment I

Do not copy verbatim from any source. Cite your sources.

Assignment 2 Part II (Groups)



Part II - Group Project (3-4 people 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 the managed resource you have chosen.
- Implement an autonomic manager to manage the 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.
- Implement a manageability interface to close the feedback loop between the managed resource and the autonomic manager.
- Make the autonomic manager policy driven.
- Demonstrate that your implementation is compliant with respect to the your chosen policy.
- Document the design and implementation of your project.

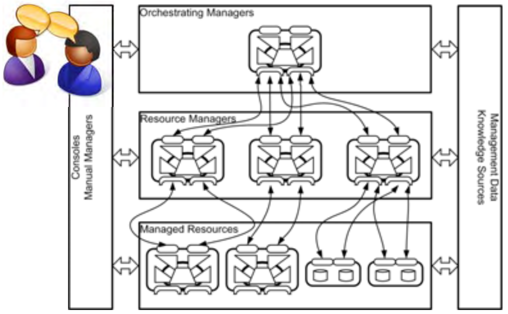
Do not copy verbatim from any source. Cite your sources.

The Continuous Evolution Problem

Devices, environments, infrastructure, web, services, business goals, user expectations, …

all evolve over time

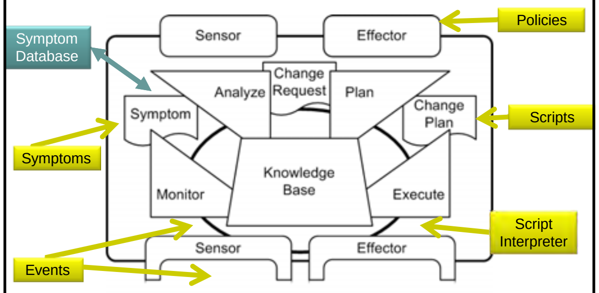
ACRA
AC Reference Architecture



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

MAPE-K Loop
Standards & Interfaces



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

Common Base Event Model CBE

- An *event*
  - An occurrence of a situation
  - Variety of forms: business, autonomic, management, tracing and logging events
  - Events encapsulate message data and constitute thus the foundation for complex distributed systems
  - Data elements of events need to be in a consistent format
    - to enable correlation
    - to facilitate effective intercommunication
- The CBE model is an event exchange standard for events exchanged in distributed applications
- The standard facilitates consistency in the elements themselves and in their format
- 3-tuple CBE element
  - Identification of the component that is *reporting* the situation
  - Identification of the component that is *affected* by the situation—may be the same as the component reporting the situation
  - The *situation* itself

Eclipse Log and Trace Analyzer

- The Eclipse Log and Trace Analyzer maps proprietary log formats into a common event model called *Common Base Event (CBE)*
- Parsers provided with the Log and Trace Analyzer map the log records from their proprietary output format to the CBE model

Event Exchange Format Standard Common Base Events (CBE)

- CBEs communicate events in a structured way
  - De facto **standard** for reporting events
  - Logging, tracking, management, or business events can all be mapped to CBEs
- CBE is an XML structure consisting of three parts:
  - Identification of the component **reporting** the situation (**reporterComponentId**)—optional; can be source
  - Identification of the component that is **affected** by the situation (**sourceComponentId**)
  - The situation itself (**situation**)

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

Event Exchange Standard: CBE Example

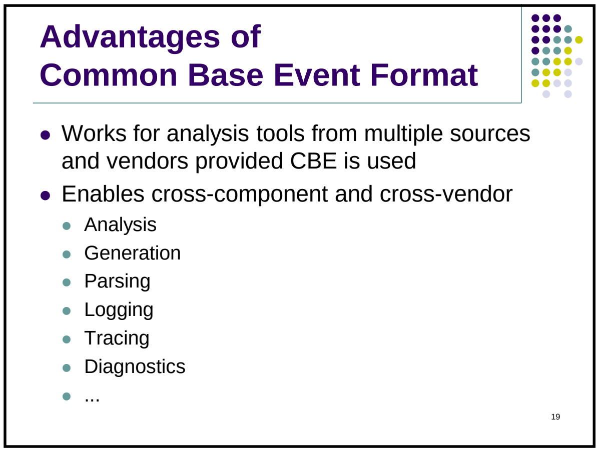
```
<CommonBaseEvent creationTime="2008-08-16T18:14:27Z"
globalInstanceId="N1FB97200C5B11D88000AB0D1D704CDE"
msg="[Fri Aug 15 18:14:27 IST 2008] ITSO001I
SampleManagedResource starting..." severity="20" version="1.0.1">
  <sourceComponentId application="ITSOSimpleApp1“
     component="ITSO Simple App#1" componentIdType="Name“
     location="server1.itso.ibm.com" locationType="IPV4“
     subComponent="ITSOSubComponent"/>
     <msgDataElement>
        <msgId>ITSO001I</msgId>
     </msgDataElement>
  <situation categoryName="StartSituation">
     <situationType xsi:type="StartSituation“
     reasoningScope="INTERNAL“
     successDisposition="SUCCESSFUL"
     situationQualifier="START INITIATED"/>
  </situation>
</CommonBaseEvent>
```

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

Generating CBEs using Eclipse

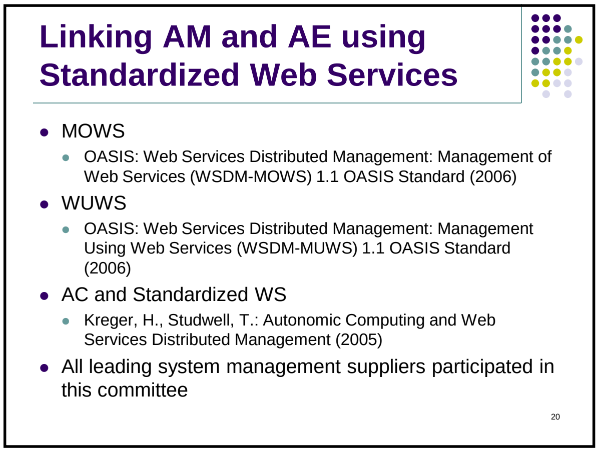
```
try {
   ISimpleEventFactory sefi = SimpleEventFactoryImpl.getInstance();
   ICommonBaseEvent cbe = sefi.createCommonBaseEvent();
   cbe.setCreationTime(System.currentTimeMillis());
   cbe.setPreferredVersion(ICommonBaseEvent.VERSION_1_0_1);

   // create a new instance of a Source Component and initialize it
   IComponentIdentification sourceComponentId =
       sefi.createComponentIdentification();
   sourceComponentId.setLocation("127.0.0.1");
   sourceComponentId.setLocationType("IPV4");
   sourceComponentId.setComponent("Ex App Server");
   sourceComponentId.setSubComponent("App Server DB");
   sourceComponentId.setComponentIdType("Application");
   sourceComponentId.setComponentType("Application Server");

   // now set source component in CBE
   cbe.setSourceComponentId(sourceComponentId);
   IConnectSituation connSituation = sefi.createConnectSituation()
   connSituation.setSuccessDisposition("UNSUCCESSFUL");
   connSituation.setSituationDisposition("AVAILABLE");
   ISituation situation = sefi.createSituation();
   situation.setCategoryName("ConnectSituation");
   situation.setSituationType(connSituation);
   cbe.setSituation(situation); // set the situation in CBE

   IMsgDataElement mde = sefi.createMsgDataElement();
   mde.setMsgId("AS005E");
   mde.setMsgIdType("AppServer");
   // add message data element to CBE
   cbe.setMsgDataElement(mde);

   // invoke manageability interface method
   // to send CBE to autonomic manager
   sendEventToManager(cbe);
}
catch (Throwable  th) {
   System.out.println("Could not create CBE: " + th); }
}
```

IBM: Autonomic Computing Toolkit Developer's Guide, Aug 2004



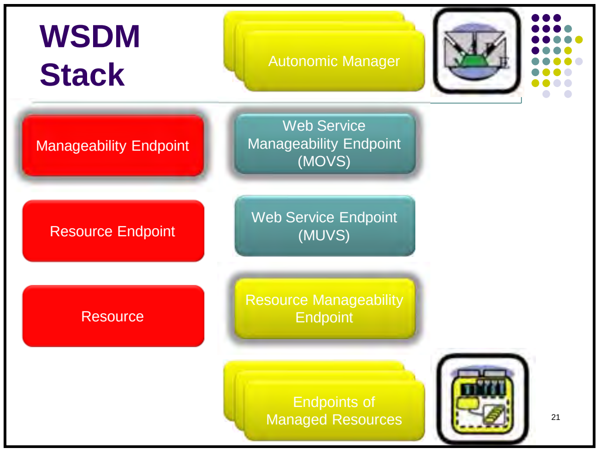
Advantages of Common Base Event Format

- Works for analysis tools from multiple sources and vendors provided CBE is used
- Enables cross-component and cross-vendor
  - Analysis
  - Generation
  - Parsing
  - Logging
  - Tracing
  - Diagnostics
  - ...



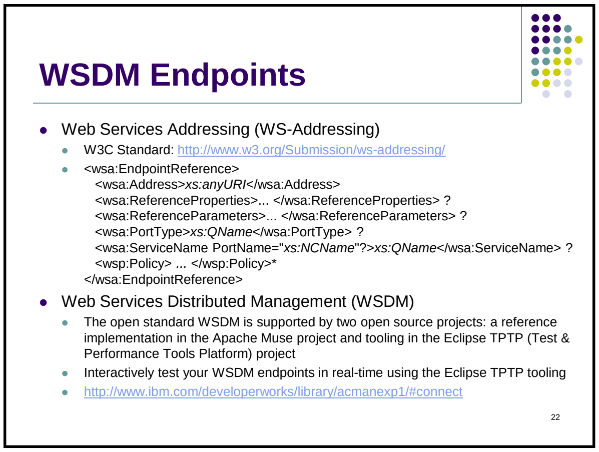
Linking AM and AE using Standardized Web Services

- 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

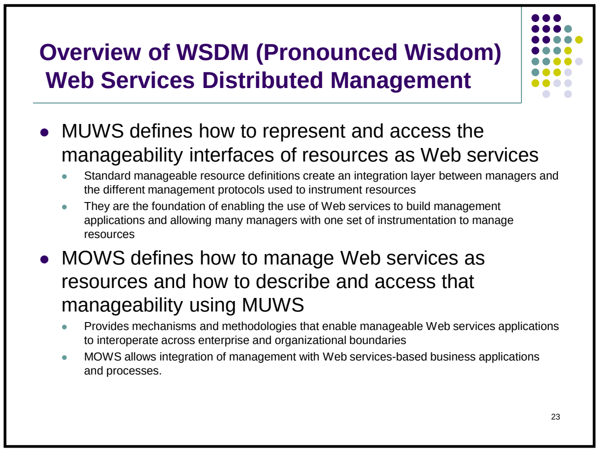
Autonomic Manager
Manageability Endpoint
Web Service Manageability Endpoint (MOVS)
Resource Endpoint
Web Service Endpoint (MUVS)
Resource
Resource Manageability Endpoint
Endpoints of Managed Resources



WSDM Endpoints

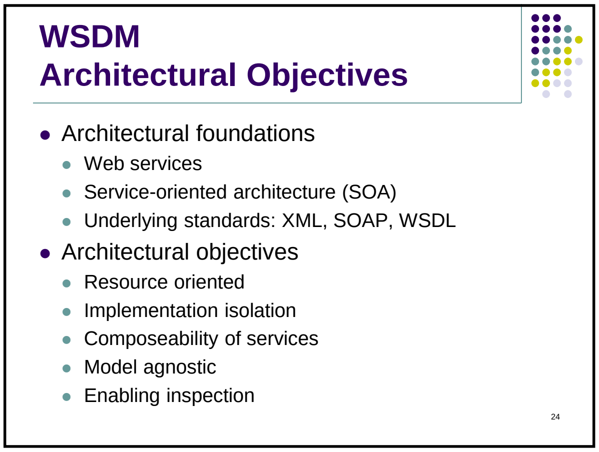
- Web Services Addressing (WS-Addressing)
  - W3C Standard: http://www.w3.org/Submission/ws-addressing/

  -
    ```
    <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

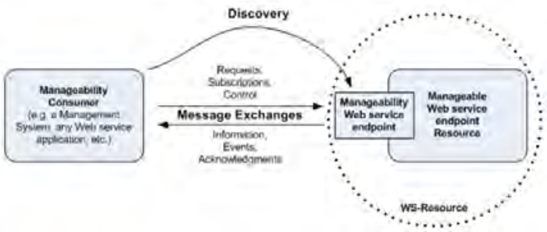
MOWS: Management Of Web Services Web Services Endpoints

- 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

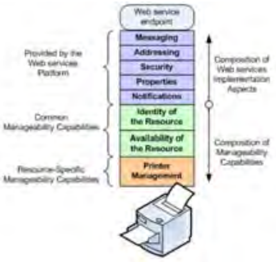
MOWS: Management Of Web Services Web Services Endpoints


Discovery
Manageability Consumer (e.g. a Management System, any Web service application, etc.)
Requests, Subscriptions, Control
Message Exchanges
Information, Events, Acknowledgments
Manageability Web service endpoint
Manageable Web service endpoint Resource
WS-Resource

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

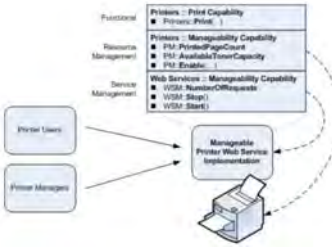
Composition of Resource Endpoint and Web Service Endpoint

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


Web service endpoint
Messaging
Addressing
Security
Properties
Notifications
Identity of the Resource
Availability of the Resource
Printer Management
Provided by the Web services Platform
Common Manageability Capabilities
Resource-Specific Manageability Capabilities
Composition of Web services Implementation Aspects
Composition of Manageability Capabilities

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

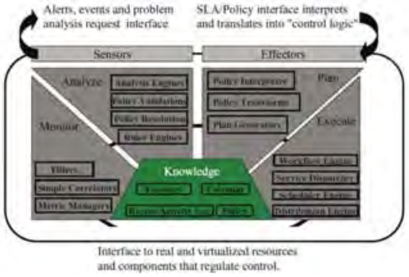

Functional
Printers :: Print Capability
Printers::Print( )
Resource Management
Printers :: Manageability Capability
PM::PrintedPageCount
PM::AvailableTonerCapacity
PM::Enable( )
Service Management
Web Services :: Manageability Capability
WSM::NumberOfRequests
WSM::Stop()
WSM::Start()
Printer Users
Printer Managers
Manageable Printer Web Service Implementation

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

Web Service Manageability Capabilities

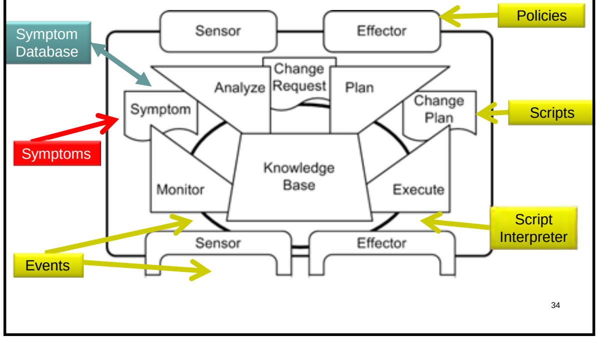
- Common manageability capabilities
- Web service endpoint manageability capabilities
- Discover web service endpoints
- Discover capabilities
- Use capabilities
- The road to autonomic computing using service-oriented architecture (SOA)

Structural Components of MAPE-K Loop



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

MAPE-K Loop Standards & Interfaces



Symptoms

- A *symptom* is a form of knowledge that indicates a possible problem or situation in the managed environment.
  - For example, "high fever" might be defined as a temperature "greater than 39 degrees Celsius"
  - The symptom is defined by the expression "temperature greater than 39 degrees Celsius" and described as "high fever"
- Symptoms are
  - Recognized in the monitor component of the MAPE-K loop
  - Used as a basis for analysis of a problem or a goal
  - Based on predefined elements—for example, definitions and descriptions in a symptoms DB
- Symptom definition
  - Expresses conditions used by the monitor to recognize the existence of a symptom
  - Specifies the unique characteristics of a particular symptom that is recognized.
- Symptoms are not just for self-healing
  - Symptoms are connected to self-healing because their primary intent is to indicate a problem
  - Symptoms can also be used as triggers for other kinds of problems
  - Virtually all kinds of problems or predictions may start due to the occurrence of a symptom

IBM: Symptoms Reference Specification Version 2.0 2006

Symptom Artifacts

- Symptom element
  - Contains all information necessary to create a new symptom occurrence
- Symptom occurrence
  - Contains the run-time information associated with a specific instance of a symptom element
  - Each occurrence basically refers to the same symptom as it is defined in the symptom element, but the context to which it is applied may vary.
- Correlation engine
  - Contains the logic used to create symptom elements
  - As input the correlation engine receives external stimuli and checks if a symptom occurrence should be created as a response.

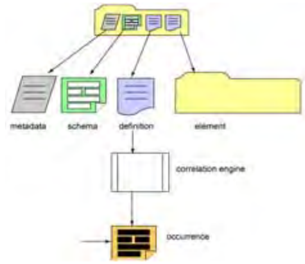
IBM: Symptoms Reference Specification Version 2.0 2006

Symptom Artifacts

- Symptom metadata
  - The generic part of the information that composes a symptom
  - It is present on all kinds of knowledge, and is used when knowledge must be treated generically, even though it is a symptom element
  - This is the "what" part of a symptom
- Symptom schema
  - The specific part of the information that composes a symptom
  - It is the template that is used when a symptom occurrence is created
  - The symptom schema contributes to the "what" part of a symptom
- Symptom definition
  - A generic piece of logic that can be used to recognize a symptom
  - As expected, this logic should be compatible with the respective correlation engine that will be used to process the symptom
  - This is the "how" part of a symptom

IBM: Symptoms Reference Specification, Version 2.0, 2006

Symptom Artifacts



M. Perazolo, IBM: Symptoms deep dive, Part 1
The autonomic computing symptoms format, Oct 2005

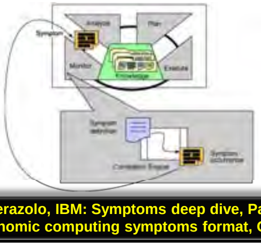
Symptom Effect Artifact

- In simple situations where no analysis or planning is performed, a symptom can be used to define the kind of reaction expected after it is recognized
- Symptom effects can also be used in an AM that implements an on-the-fly strategy for creating change requests
- Symptom effect artifact could be
  - An action to be performed in a manageable resource
  - A human readable recommendation
  - Something simple such as running a script or a piece of code
- The current symptom specification defines only two forms of effect
  - **Recommendation:** A textual representation of what an operator should do to fix the problem associated with a particular symptom
  - **Action:** A piece of code that defines tasks and procedures used to fix the problem associated with a particular symptom

M. Perazolo, IBM: Symptoms deep dive, Part 1
The autonomic computing symptoms format, Oct 2005

Symptoms in MAPE-K Loop

- Symptoms are recognized by a correlation engine in the monitor
- After recognition a new symptom occurrence is created
- The symptom occurrence is then passed from the monitor to the analysis part of the MAPE-K loop



M. Perazolo, IBM: Symptoms deep dive, Part 1
The autonomic computing symptoms format, Oct 2005

Symptom Metadata

Property	Description
Identification	Identifies a symptom uniquely within the MAPE-K loop A unique alphanumeric id
Versioning	Contains the change history associated with the symptom
Annotation	Describes the symptom in a human readable form to explain different characteristics of the symptom
Location	Tells where the authoritative version of this symptom resides and points to the original K-source that contains the symptom
Scope	The manageable resource type a symptom can be applied to At run time, the scope property will also contain the context associated with the symptom occurrence (e.g., the instance of the manageable resource type that is the root cause of the problem or indication defined by the symptom).
Lifecycle	A run-time property containing the current state associated with a symptom occurrence In the case of symptoms, the states are: created, building, analyzed, planning, executing, scheduled, completed, expired, and fault.

Symptom Schema

Attribute	Description
Description	Explains in a human readable form what the symptom is about. Describes the kinds of problems or situations associated with the symptom when a symptom occurrence is recognized by an AM.
Example	Shows in a human-readable form an example of a problem or situation where the symptom is likely to occur.
Solution	Shows in a human-readable form a possible solution for the problem or situation described by the example attribute.
Reference	Contains a URL associated with the symptom that lets a user get the latest information associated with that symptom from the Web.
Type	Contains the type associated with the symptom occurrence. It ultimately equates to a symptom category that enables you to organize multiple symptoms in a common taxonomy of symptoms.
Probability	Denotes the probability or certainty associated with the problem or situation indicated by a symptom occurrence.
Priority	Denotes the priority of a symptom occurrence in relation to other symptoms with the same scope.

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

What is the role of a symptoms database?
How general is the notion of a symptom?

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

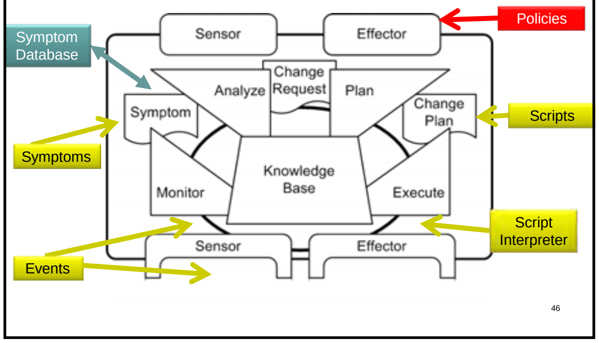
Symptom name	Symptom description	Symptom definition	Symptom recommendations
Configuration unavailable	Some configuration information for the resources associated with this symptom was not found	Filter pattern: event(configuration_not_found)	Alert administrator and flag service provided by resource as "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



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)