Dependable Systems

Trends in Software Dependability

Dr. Peter Tröger

Autonomic Computing

- Initiative started by IBM in October 2001 with manifesto
  - Main obstacle for future IT is looming software complexity crisis
  - Applications with tens of millions lines of code, require skilled personal
  - System complexity approaches limit of human capability
  - System become more interconnected and diverse
- Create self-managing computer systems capable of coping with growing complexity, based on high-level objectives from administrators
- New paradigm for design and implementation of systems
- Term derives from body‘s autonomic nervous system, which controls key functions without conscious awareness

Credo

- Exhibit basic fundamentals - from a **user perspective**
  - **Flexible** - Sift data with a platform- and device-agnostic approach
  - **Accessible** - ‚Always on‘ nature
  - **Transparent** - Adapt to user needs
    - Perform tasks without involving the user into operational details
- Minimize human interference
- Policies (goals or objectives) govern the behavior of **intelligent control loops**

8 Elements [IBM]

- **Characteristics** of an autonomic system
  - Computing system needs to **know itself**, having a system identity
    - Detailled knowledge of components, their current status, and capacity
    - Knowledge about connections to other systems
    - Knowledge about extend of owned resources, those it can borrow or lend, those that can be shared or should be isolated
    - Goal: To govern itself
  - Computing system must perform **self-configuration** automatically
    - Under varying and (future) unpredictable conditions
    - Setup must occur automatically, constant dynamic adjustment to environment
    - Example: Installing software when a pre-requisite is missing

8 Elements [IBM]

- Computing system always looks for ways for **self-optimization**
  - System never settles for status quo, tries to achieve optimum of predefined goals with minimum resources
  - Monitor important parts, fine-tune workflow for best functioning
  - Example: Adjust workload according to available resources
- Computing system must aim at automated **self-healing**
  - Discover (potential) problems, find alternate way of using resources
  - Recover from such extraordinary events that might cause malfunction
  - Example: Correcting a configured path to correctly load software

8 Elements [IBM]

- Computing system should act in an **adaptive** fashion
  - Must know its environment and the surrounding context activity
  - Find and generate rules for how best to interact with neighbours
  - Changing both itself and its environment
- Computing system **cannot be a proprietary solution**
  - Autonomic system cannot exist in a hermetic environment - open standards
  - Independent in its ability to manage itself, but must function in a heterogeneous world
  - Implementation of open standards

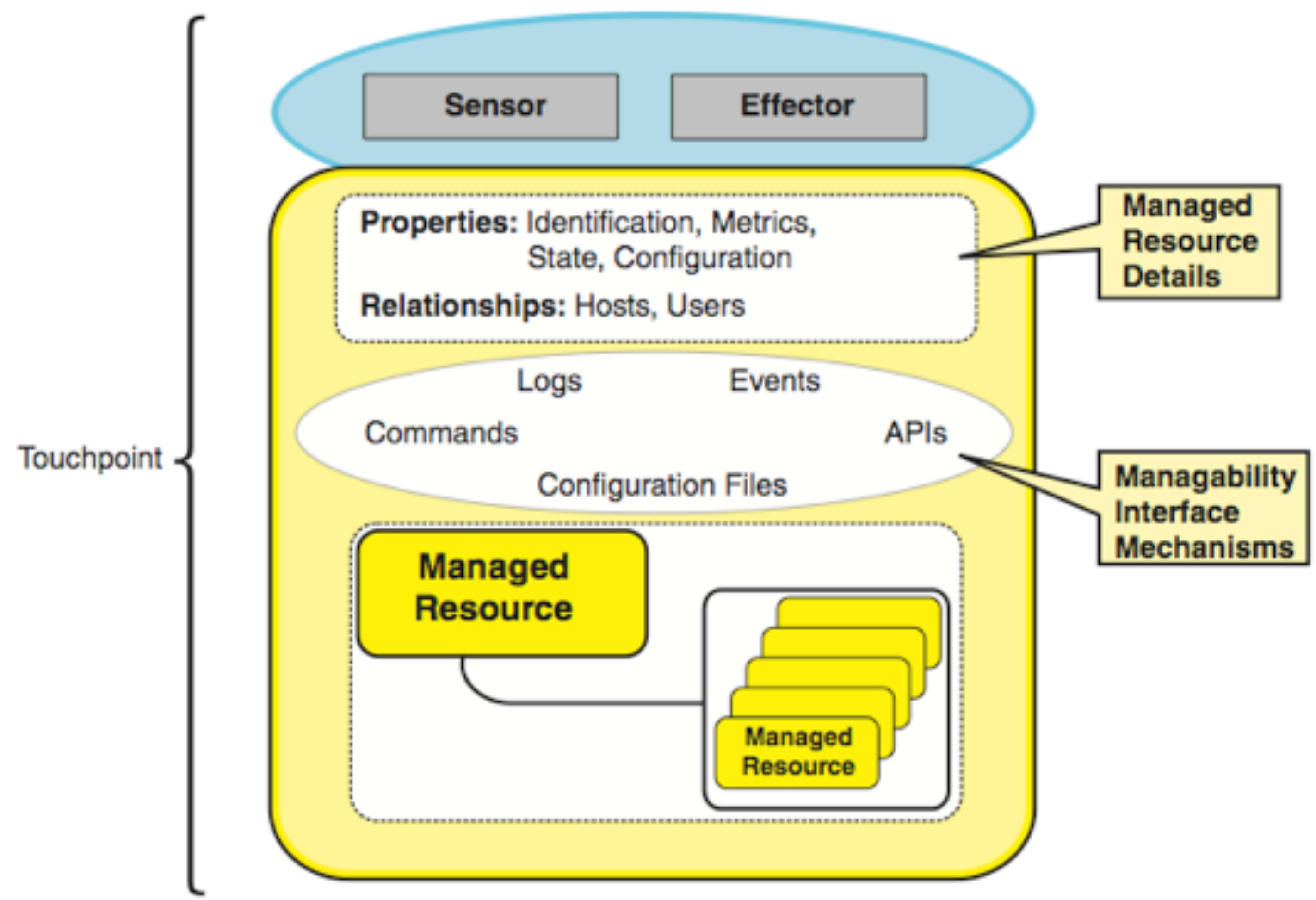
8 Elements [IBM]

- Computing system must be an expert in **self-protection**
  - Detect, identify and protects itself against arbitrary attacks
  - Pro-active and reactive behavior
  - Example: Take resource offline if intrusion attempt is detected
- Computing system must **keep complexity hidden**
  - Marshal IT resources to shrink the gap between
    - Business / user goals and
    - IT implementation necessary to achieve those goals
  - Do not involve the user in this activity

Concepts

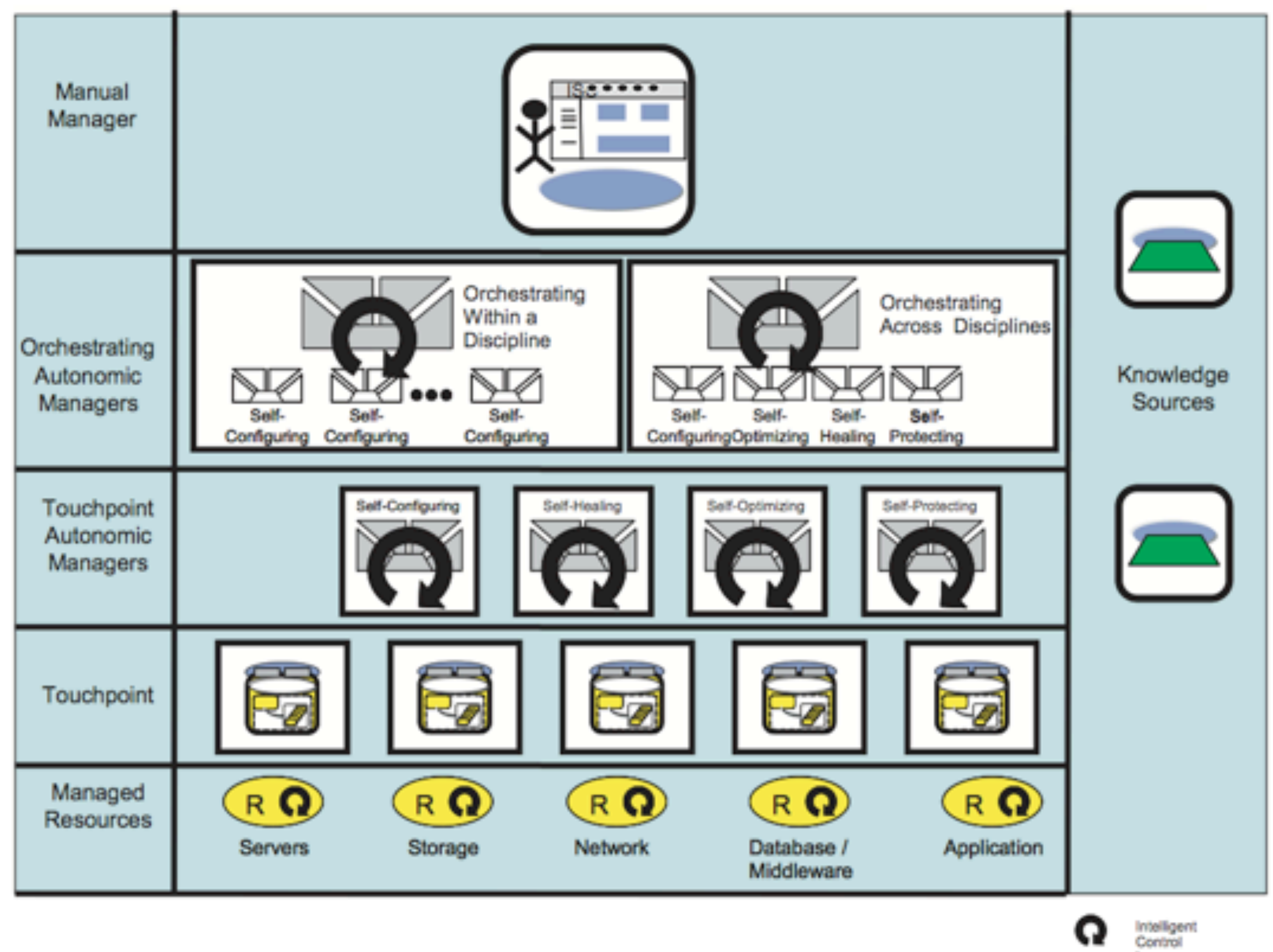
- **CHOP Features**
  - Self-Configuration
  - Self-Healing
  - Self-Optimization
  - Self-Protection
- **MAPE Loop**
  - Monitor -> Analyze -> Plan -> Execute
  - Base for autonomic management by a control loop concept

Managed Resource



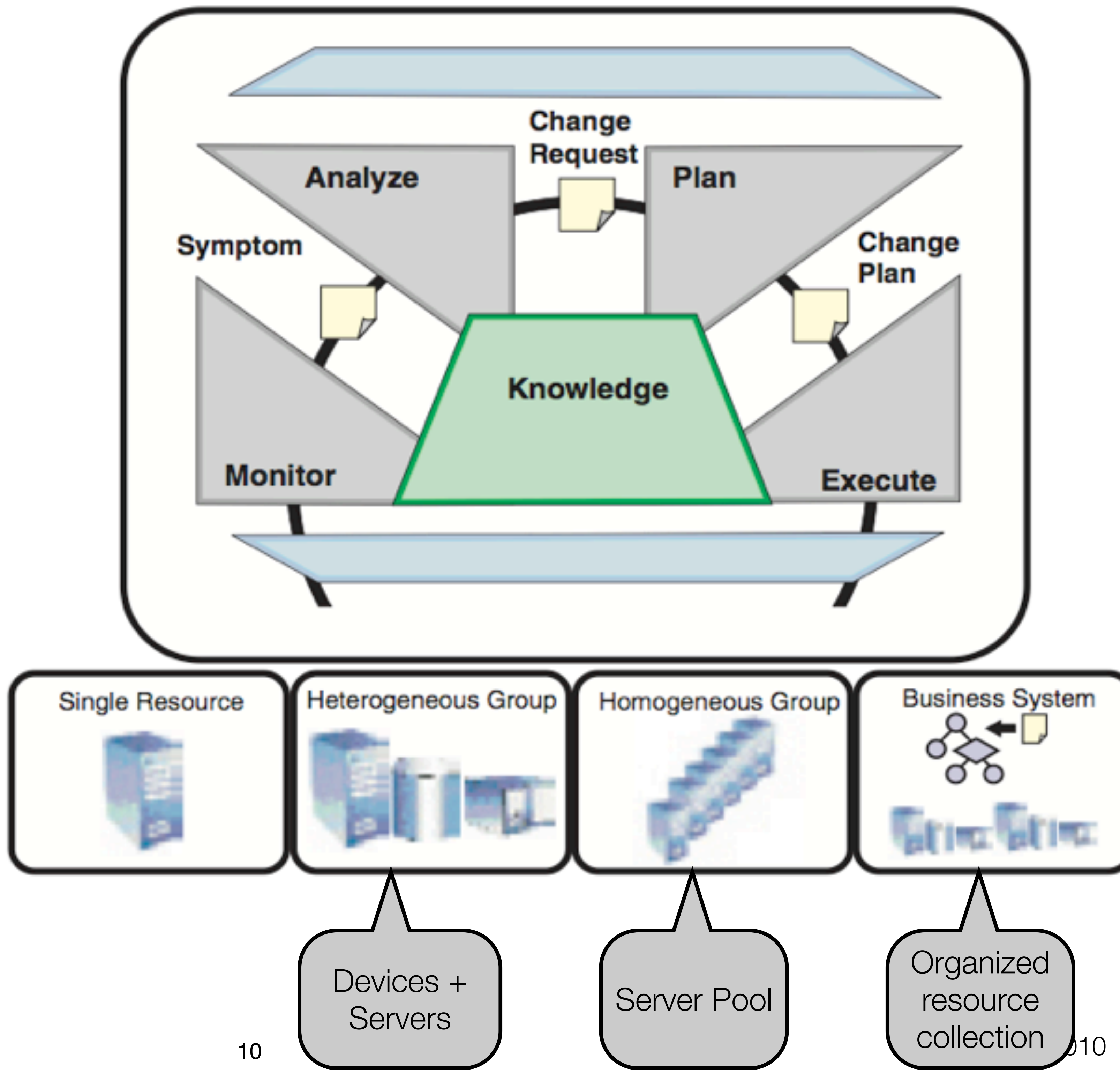
System Layers

- Autonomic Manager - Manages other software and hardware, using a control loop
- Touchpoint - Interface to an instant of a managed resource (OS, server, hardware)
  - Includes manageability interface for monitoring and control0
  - Also expose sensor and effector
- Event - Significant change in system state
- Sensor - Exposes information about managed resource state and state transitions
- Effector - Enables state changes
- Interaction based on *Enterprise Service Bus*

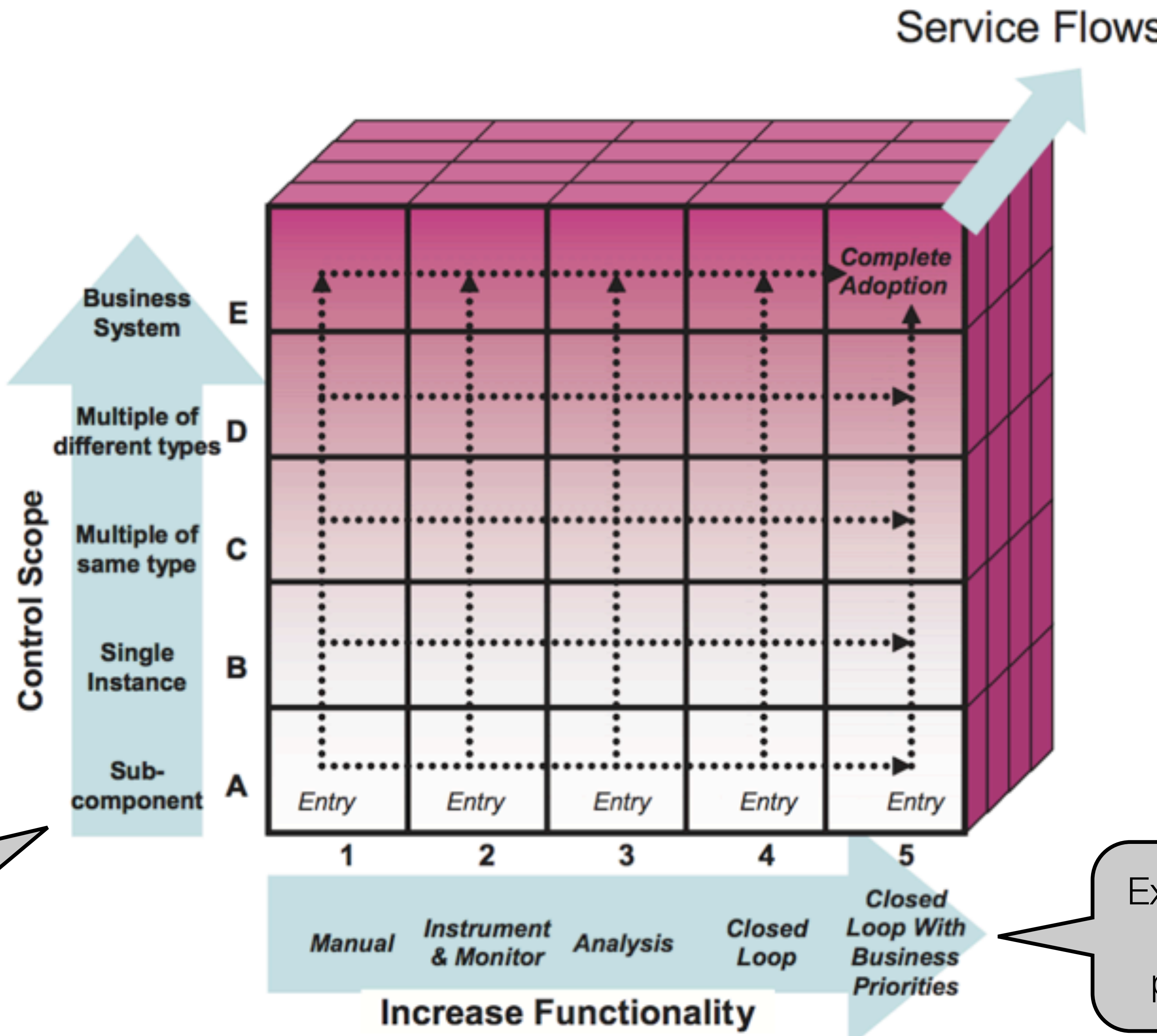


MAPE Cycle in the Autonomic Manager

- Administrators can decide to realize only parts of the control loop
  - Evolutionary process
- **Monitor** - Correlates sensor values into symptoms
- **Analyze** - Determine need for some change
- **Plan** - Creates or selects procedure to enact resource alteration
- **Execute** - Carry out the actions, update internal knowledge



Autonomic Computing Adoption Model



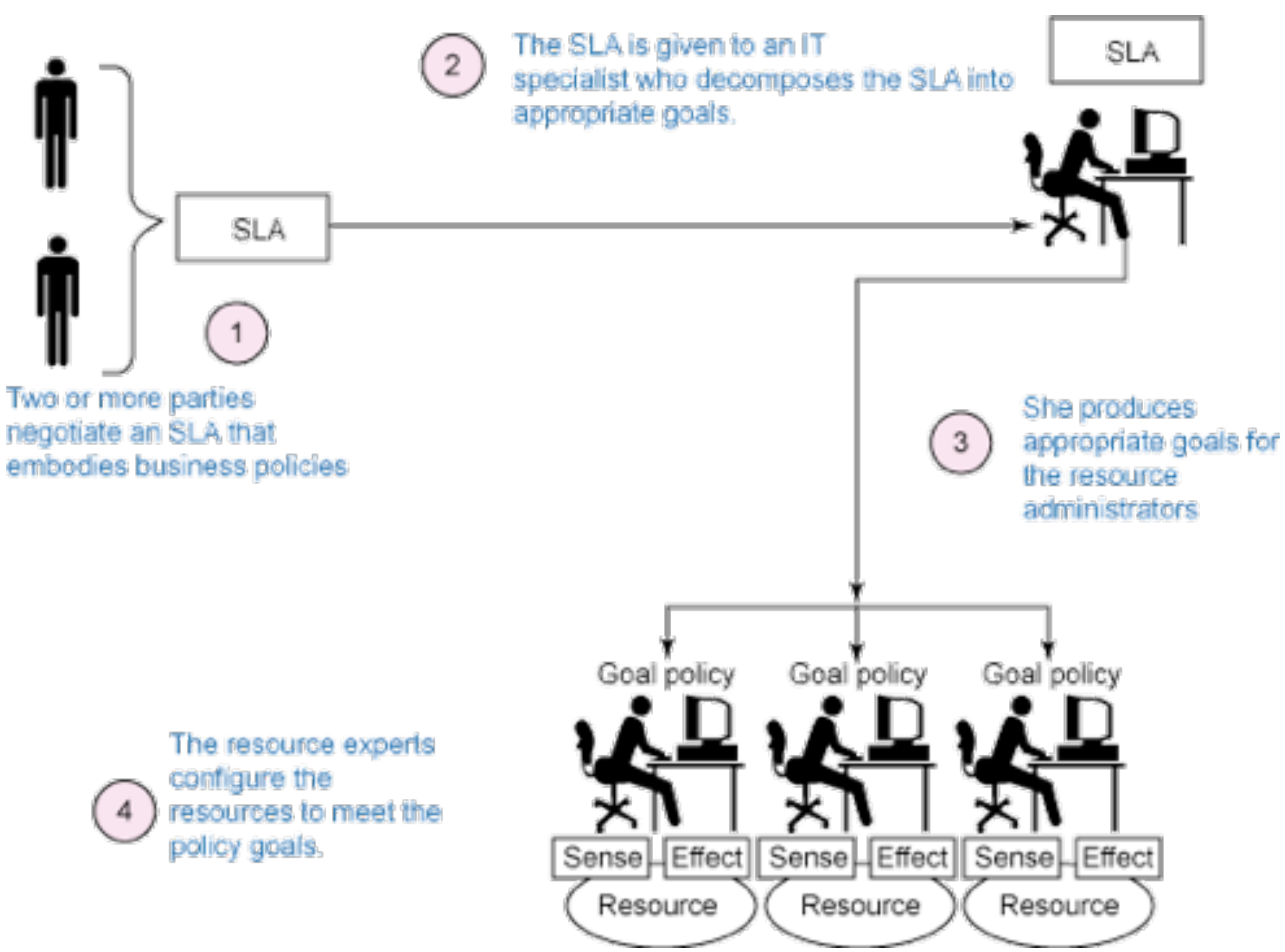
What is managed

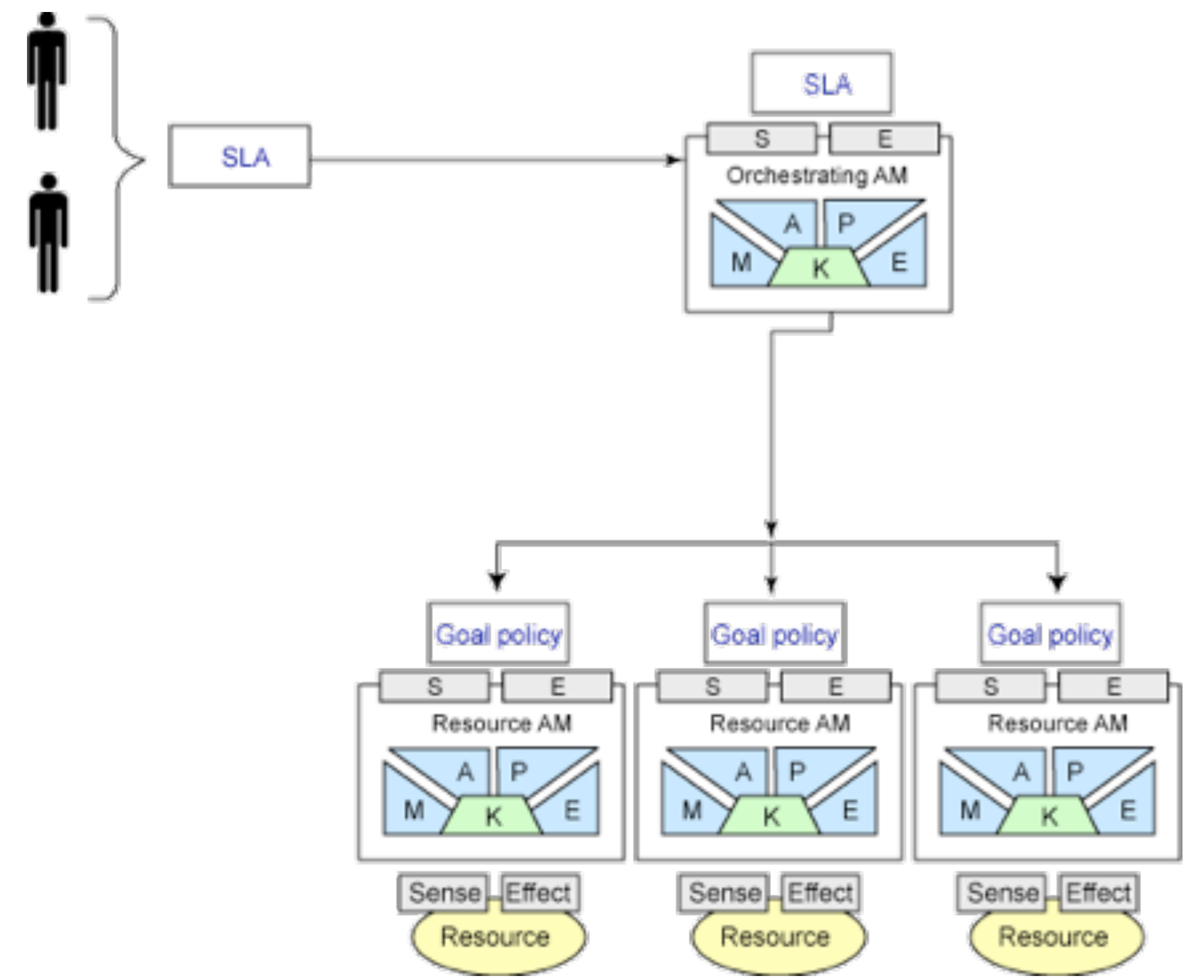
Extend of automation for the IT and business process management

Standards - Some Examples

- Sensors and effectors should confirm to standards
  - Distributed Management Task Force (DMTF)
    - Common Information Model (CIM)
    - Web Services Common Information Model (WS-CIM)
  - Internet Engineering Task Force (IETF)
    - Policy - Core Information Model (RFC3060)
    - Simple Network Management Protocol (SNMP)
  - Java Community Process (JCP)
    - Java Agent Services (JSR87)
    - Java Management Extensions (JSR3, JMX)

Level 5 - Policy Example





Level 5 - SLA Example

- Agreed SLA
  - *From 9:00 a.m. - 5:00 p.m., users of the trading application "MyApp" will not average more than 1 second response time*
  - *The application "MyApp" is always available*
  - *I can run reports without interrupting MyApp*
- System administrator derives according goal policy
  - *Goal (from SLA): On average, users will not wait more than 1 second*
  - *Policy Scope: trading application "MyApp"*
  - *Policy Condition: 9:00 a.m. to 5:00 p.m.*
  - *Policy Decision: Average Response Time < 0.9 second*
  - *Policy Business Value: 500*

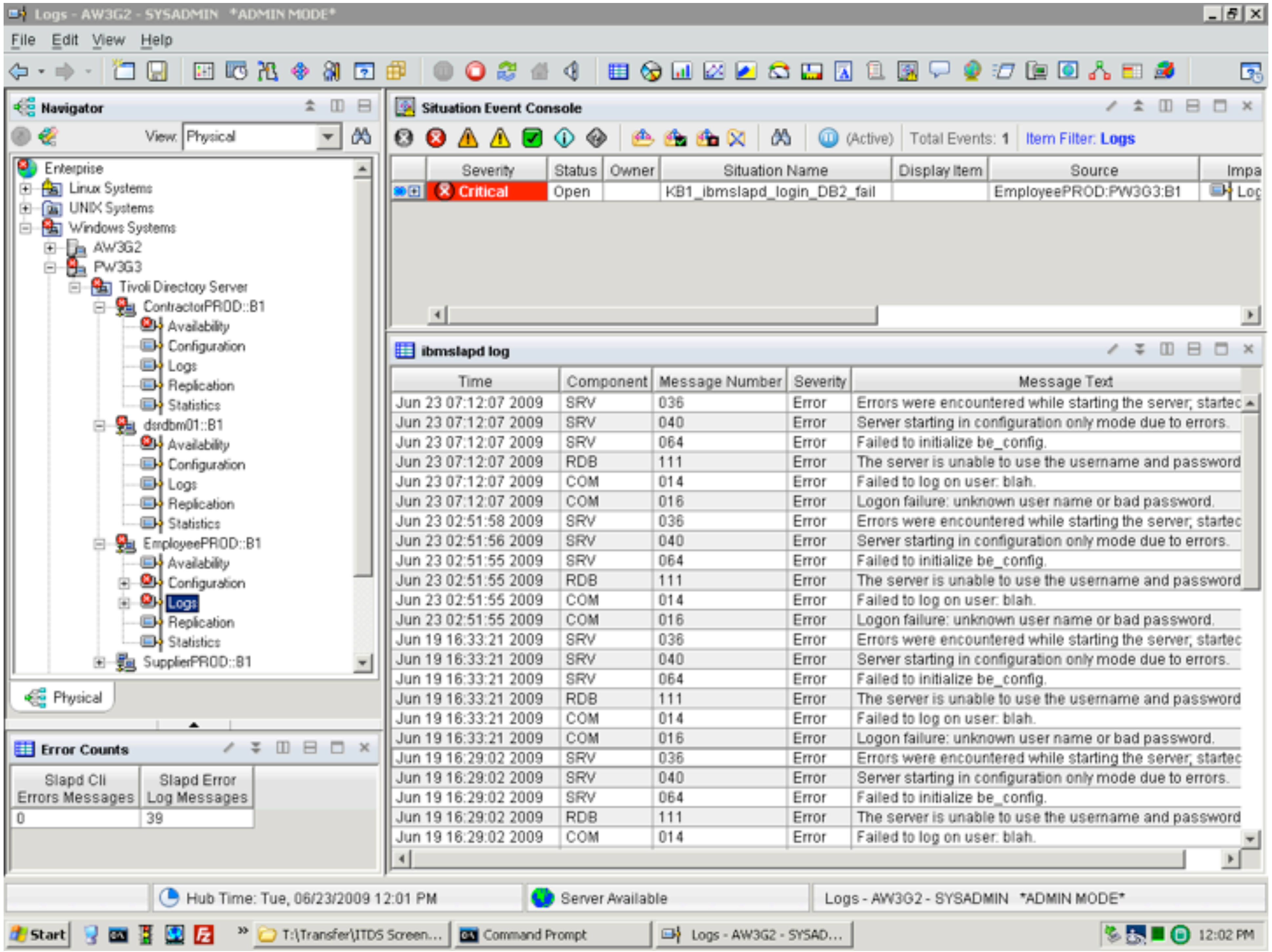
Level 5 - SLA Example

- Individual goal policies for resources derived
- *Policy Scope: Storage*
  - *Policy Condition: Average CPU utilization > 66%*
  - *Policy Decision: Increase cache allocation for MyApp by 10%*
  - *Policy Business Value: 500*
- *Policy Scope: Application*
  - *Policy Condition: Average Response Time > 200 ms*
  - *Policy Decision: Reduce priority of all low priority queries*
  - *Policy Business Value: 600*
- *Policy Scope: Network*
  - *Policy Condition: Network Response Time > 100 ms*
  - *Policy Decision: Increase Quality of Service parameters for MyApp's IP address*
  - *Policy Business Value: 400*

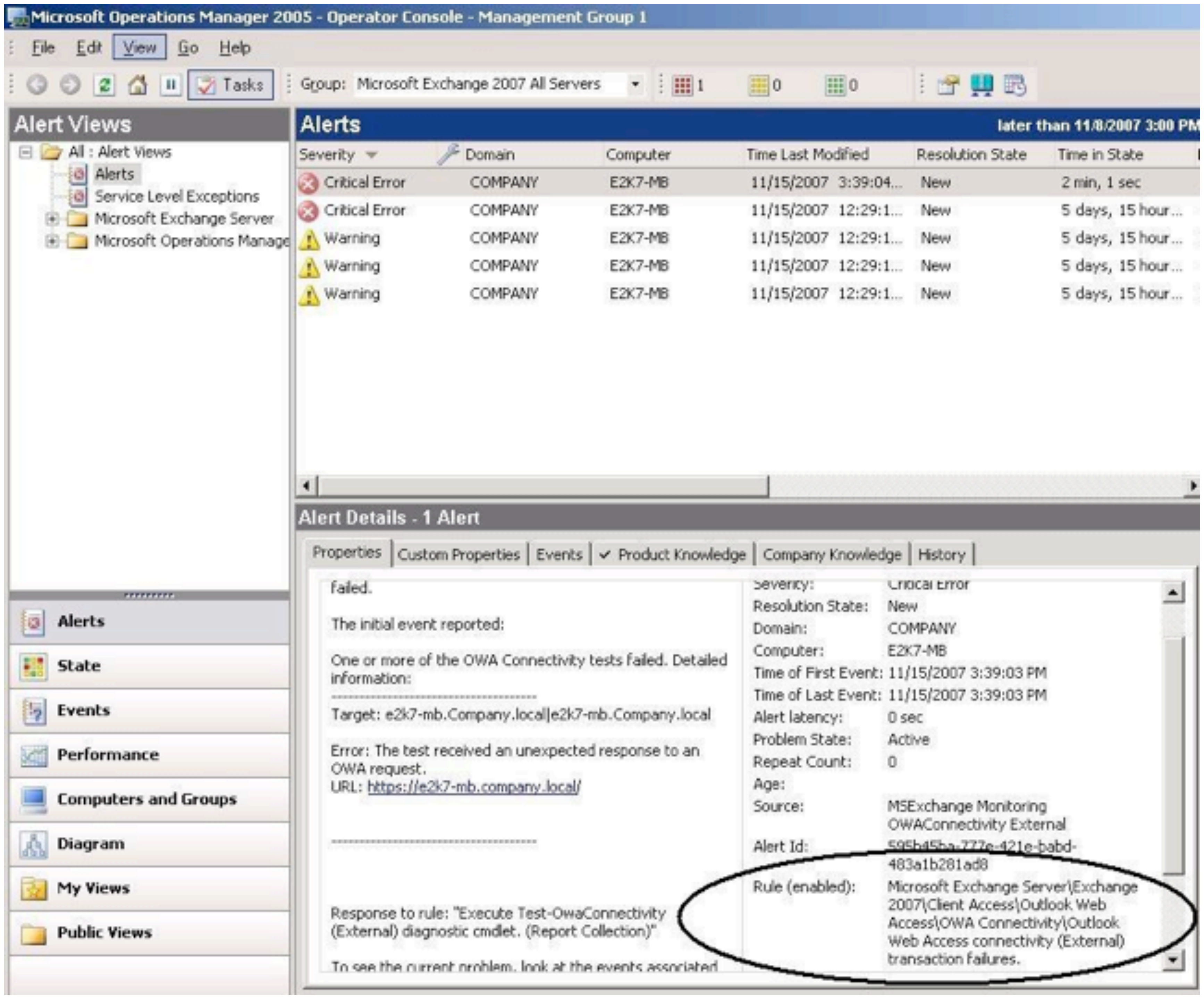
Real Projects

- Network Solutions (domain registration company)
  - Tivoli Management Framework
    - Adaptors on resources convert to common log format
    - Self-recovery for server cluster - automated startup / shutdown
    - Close connection to other IBM products
  - Tivoli Enterprise Console
    - Beyond simple filtering, allows root cause analysis
    - Pre-configured rules for event management
- Comparable activities with competitors

IBM Tivoli


Logs - AW3G2 - SYSADMIN *ADMIN MODE*
File Edit View Help
Navigator
View: Physical
Enterprise
Linux Systems
UNIX Systems
Windows Systems
AW3G2
PW3G3
Tivoli Directory Server
ContractorPROD::B1
Availability
Configuration
Logs
Replication
Statistics
dsrdbm01::B1
Availability
Configuration
Logs
Replication
Statistics
EmployeePROD::B1
Availability
Configuration
Logs
Replication
Statistics
SupplierPROD::B1
Physical
Error Counts
Slapd Cli Errors Messages
Slapd Error Log Messages
0
39
Situation Event Console
(Active) Total Events: 1 Item Filter: Logs
Severity Status Owner Situation Name Display Item Source Impa
Critical Open KB1_ibmslapd_login_DB2_fail EmployeePROD:PW3G3:B1 Loç
ibmslapd log
Time Component Message Number Severity Message Text
Jun 23 07:12:07 2009 SRV 036 Error Errors were encountered while starting the server; startec
Jun 23 07:12:07 2009 SRV 040 Error Server starting in configuration only mode due to errors.
Jun 23 07:12:07 2009 SRV 064 Error Failed to initialize be_config.
Jun 23 07:12:07 2009 RDB 111 Error The server is unable to use the username and password
Jun 23 07:12:07 2009 COM 014 Error Failed to log on user: blah.
Jun 23 07:12:07 2009 COM 016 Error Logon failure: unknown user name or bad password.
Jun 23 02:51:58 2009 SRV 036 Error Errors were encountered while starting the server; startec
Jun 23 02:51:56 2009 SRV 040 Error Server starting in configuration only mode due to errors.
Jun 23 02:51:55 2009 SRV 064 Error Failed to initialize be_config.
Jun 23 02:51:55 2009 RDB 111 Error The server is unable to use the username and password
Jun 23 02:51:55 2009 COM 014 Error Failed to log on user: blah.
Jun 23 02:51:55 2009 COM 016 Error Logon failure: unknown user name or bad password.
Jun 19 16:33:21 2009 SRV 036 Error Errors were encountered while starting the server; startec
Jun 19 16:33:21 2009 SRV 040 Error Server starting in configuration only mode due to errors.
Jun 19 16:33:21 2009 SRV 064 Error Failed to initialize be_config.
Jun 19 16:33:21 2009 RDB 111 Error The server is unable to use the username and password
Jun 19 16:33:21 2009 COM 014 Error Failed to log on user: blah.
Jun 19 16:33:21 2009 COM 016 Error Logon failure: unknown user name or bad password.
Jun 19 16:29:02 2009 SRV 036 Error Errors were encountered while starting the server; startec
Jun 19 16:29:02 2009 SRV 040 Error Server starting in configuration only mode due to errors.
Jun 19 16:29:02 2009 SRV 064 Error Failed to initialize be_config.
Jun 19 16:29:02 2009 RDB 111 Error The server is unable to use the username and password
Jun 19 16:29:02 2009 COM 014 Error Failed to log on user: blah.
Hub Time: Tue, 06/23/2009 12:01 PM
Server Available
Logs - AW3G2 - SYSADMIN *ADMIN MODE*
Start
T:\Transfer\ITDS Screen...
Command Prompt
Logs - AW3G2 - SYSAD...
12:02 PM

Microsoft MOM



Solaris Fault Manager

External Slide Set

Solaris 10
Predictive Self-Healing:
Fault Management

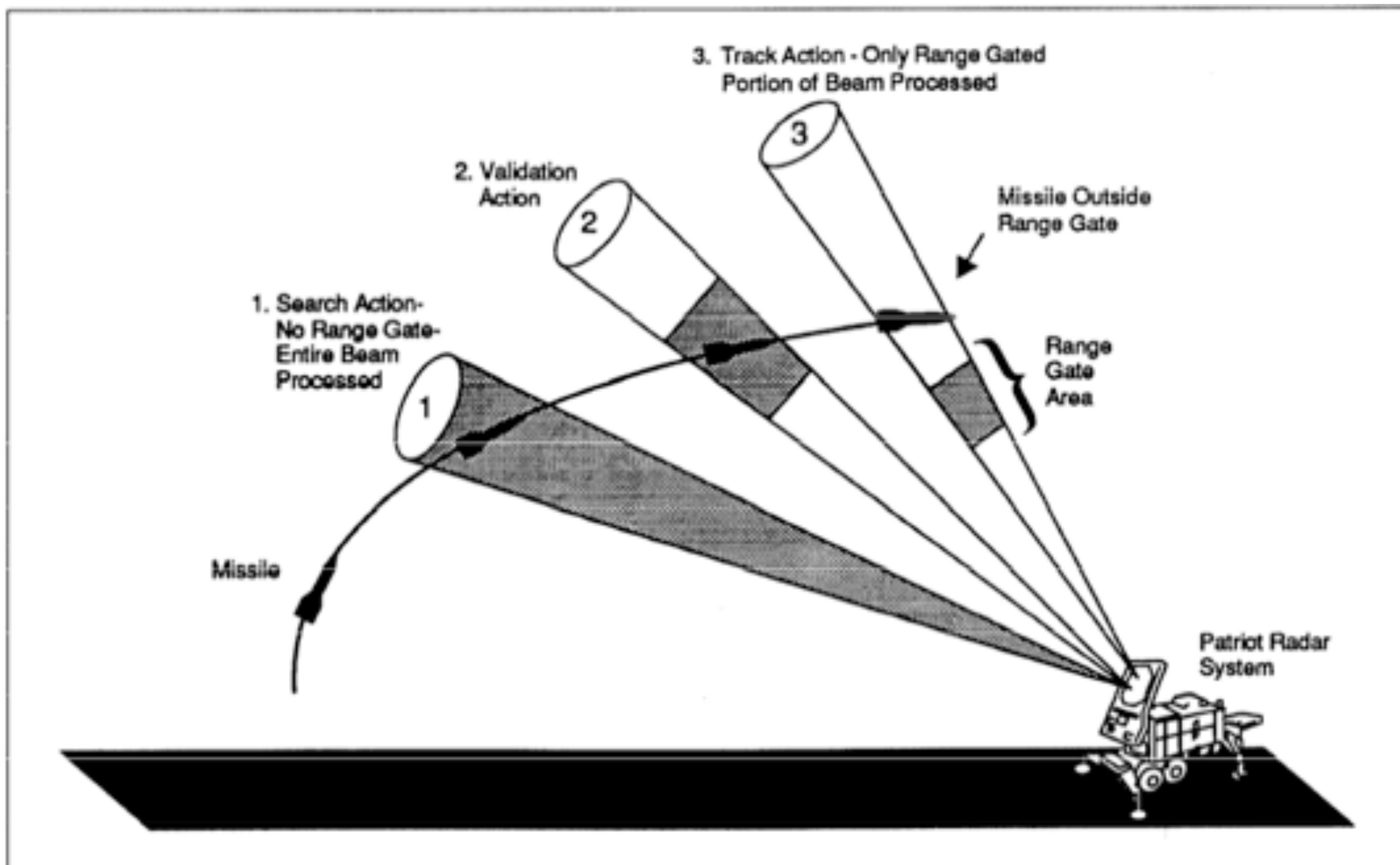
Mike Shapiro
mws@sun.com http://blogs.sun.com/mws/
Solaris Kernel Development, Sun Microsystems

Software Rejuvenation

- Software faults
  - Testing and debugging aims at **Bohrbugs**
  - **Heisenbugs**: Non-deterministic manifestation, depend on rare states and timing
  - Some problems come from **software aging**
    - Data corruption, numerical error accumulation, OS resource exhaustion
    - Error conditions accumulate over time
    - Example faults: Memory leaks, algorithmic data corruption, fragmentation
    - Example errors: Crash, application hang, performance degradation, transient problems, computational failures due to accumulated non-urgent issues

Example: Patriot Missile Launcher

- Mobile missile launcher, designed for a few hours of operation
- February 1991 - Battery in Dharan, Saudi Arabia failed to intercept Scud missile
  - Software aging problem in system‘s weapon control computer
    - Target velocity and time demanded as real values, stored as 24-bit integer
  - Inaccurate tracking computation due to overlong operation (> 100 hours)
  - Modified software reached the base one day after the accident
  - Missile launcher was never designed for Scud defense operation



Approaches

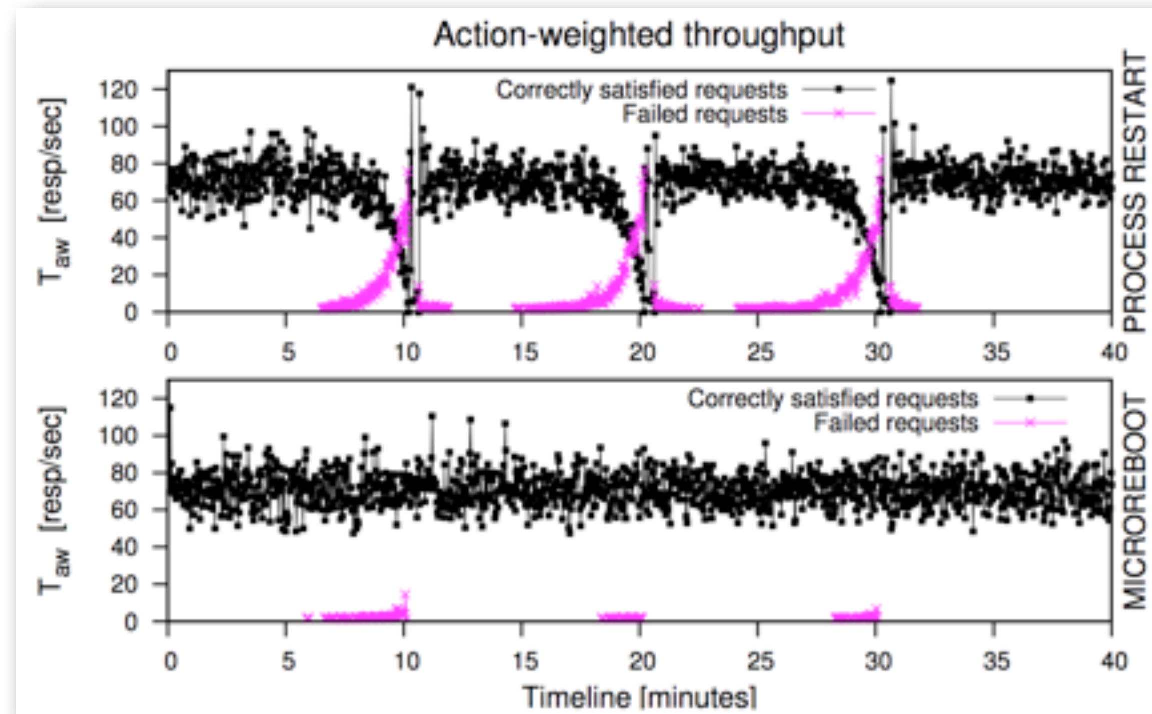
- Use time redundancy to deal with transient software bugs
    - Restart, rollback, roll-forward, progressive retry, occasional reboot
- **Proactive fault management**
    - Postpone and or prevent crashes (decrease failure rate) and prevent performance degradation (increase service rate)
- **Software rejuvenation**
    - Stop software regularly, clean internal state and / or environment, restart it
    - Counteracts aging problem - resources are freed, accumulated errors are gone
    - Several approved cleaning techniques - Garbage collection, defragmentation, table flushing, graceful restart
    - Major research issue in optimal rejuvenation interval, due to overhead
    - Different escalation levels: Process restart, application restart, node restart

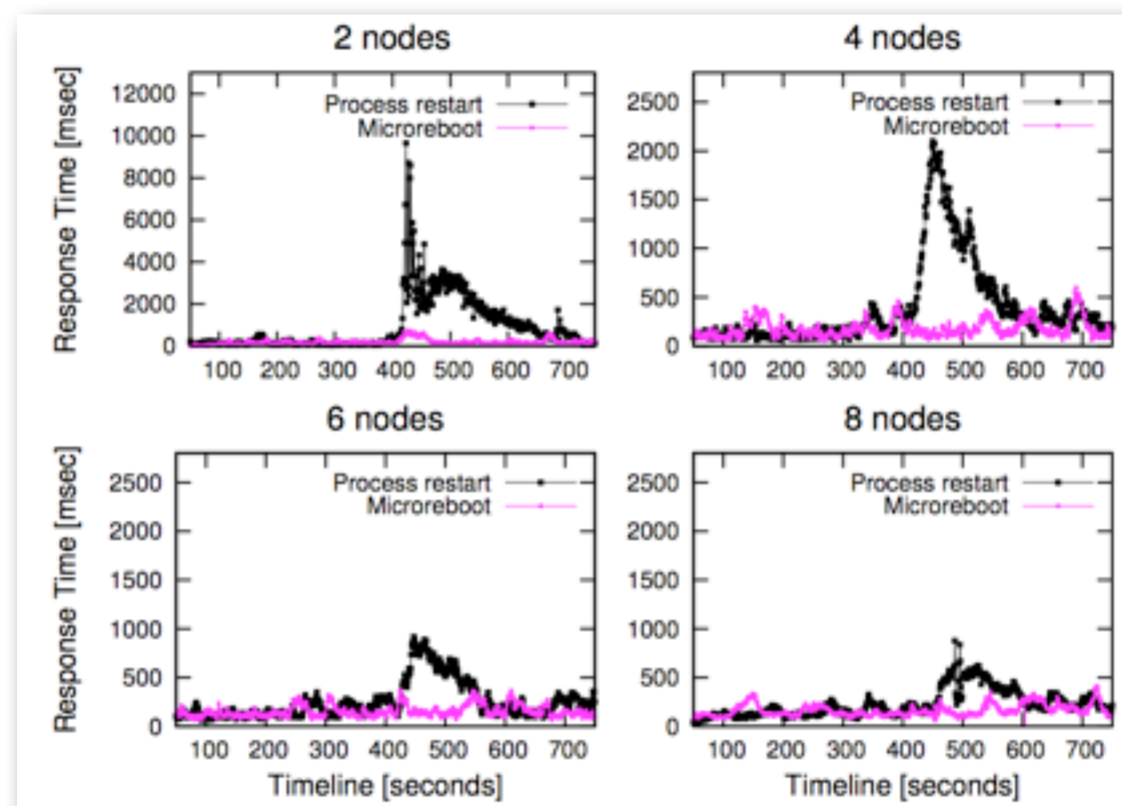
Example: Microreboot [Candea et al.]

- Idea: Establish micro-reboots for Java EE beans
    - Implemented in Java EE, fault model from real-world feedback
    - Evaluated on auction system, all state externalized
    - Micro-reboot of EJB and its transitive closure of deployment dependencies
- Based on concept of **crash-only software**
    - Programs that can be safely crashed and recover quickly every time
    - Fine-grained components with explicit boundaries
    - State segregation
    - Retryable requests - Callers should be able to gracefully recover
    - Resources should be leased - CPU time, network bandwidth, request TTL

Example: Microreboot [Candea et al.]

Injected Fault	Type	Reboot level	+
Deadlock		EJB	
Infinite loop		EJB	
Application memory leak		EJB	
Transient exception		EJB	
Corrupt primary keys	set null	EJB	
	invalid	EJB	
	wrong	EJB	≈
Corrupt JNDI entries	set null	EJB	
	invalid	EJB	
	wrong	EJB	
Corrupt transaction method map	set null	EJB	
	invalid	EJB	
	wrong	EJB	≈
Corrupt stateless session EJB attributes	set null	unnecessary	
	invalid	unnecessary	
	wrong	EJB+WAR	≈
Corrupt data inside FastS	set null	WAR	
	invalid	WAR	
	wrong	WAR	≈
Corrupt data inside SSM	corruption detected via checksum; bad object automatically discarded		
Corrupt data inside MySQL	database table repair needed		
Memory leak outside application	intra-JVM	JVM/JBoss	
	extra-JVM	OS kernel	
Bit flips in process memory		JVM/JBoss	≈
Bit flips in process registers		JVM/JBoss	≈
Bad system call return values		JVM/JBoss	





Recovery-Oriented Computing

External Slide Set

Recovery Oriented Computing

Dave Patterson
University of California at Berkeley
Patterson@cs.berkeley.edu

http://roc.CS.Berkeley.EDU/

September 2001