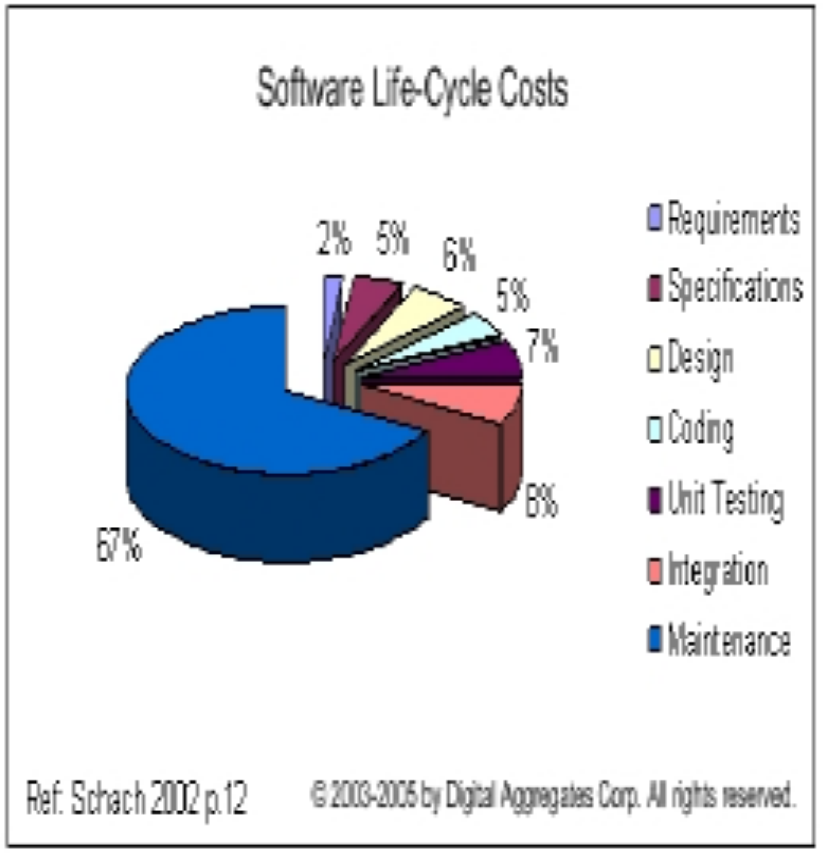
# Autonomic Computing

# Introduction

- Administration of individual systems is increasingly difficult
  - 100s of configuration, tuning parameters
- Heterogeneous systems are becoming increasingly connected
  - Integration becoming ever more difficult
- Engineers can't intricately plan interactions among components
  - Increasingly dynamic; more frequently with unanticipated components
- The burden must be assumed at run time
  - But human system administrators can't assume it
    - 40% outages due to operator error

# Introduction

The growing complexity of the IT infrastructure threatens to undermine the benefits of information technology



# Introduction

- In 2001, IBM released a manifesto
  - Software complexity crisis
  - Trying to reduce the system complexity
    - Improve system performance
    - Provide suitable QoS
    - Optimize the existing resources

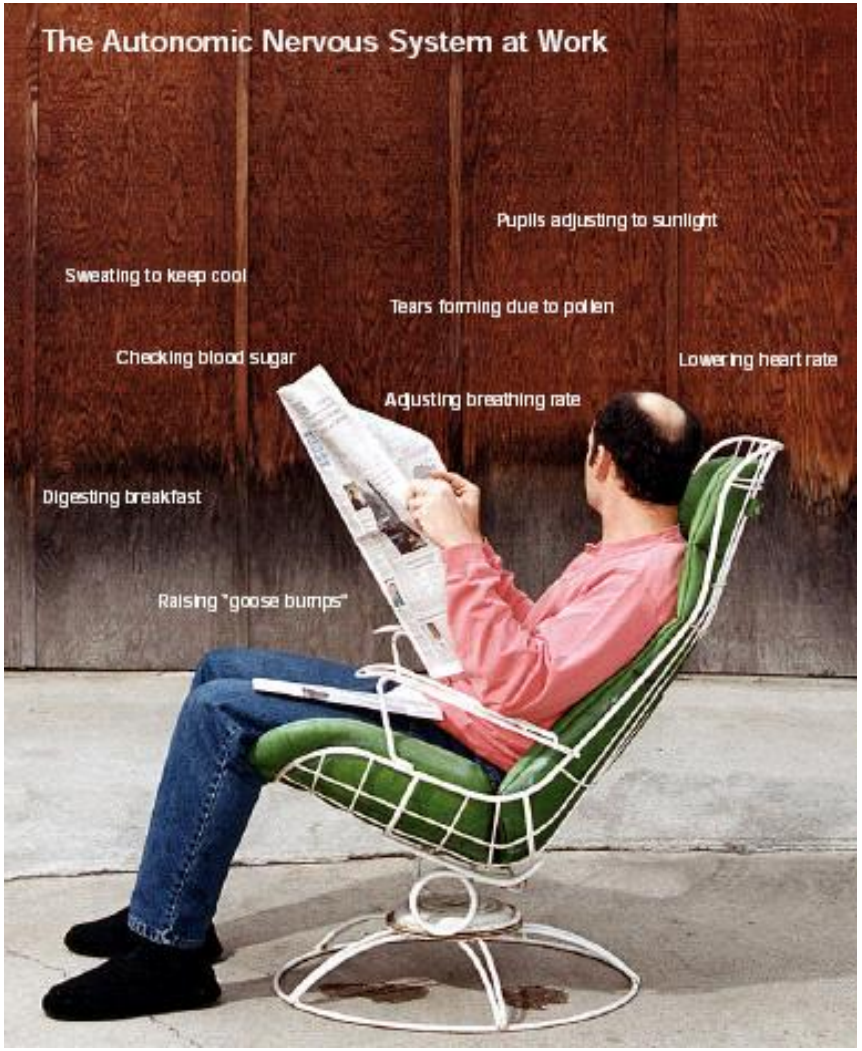
# Autonomic Computing

- **Autonomic:** pertaining to an on demand operating environment that responds automatically to problems, security threats, and system failures

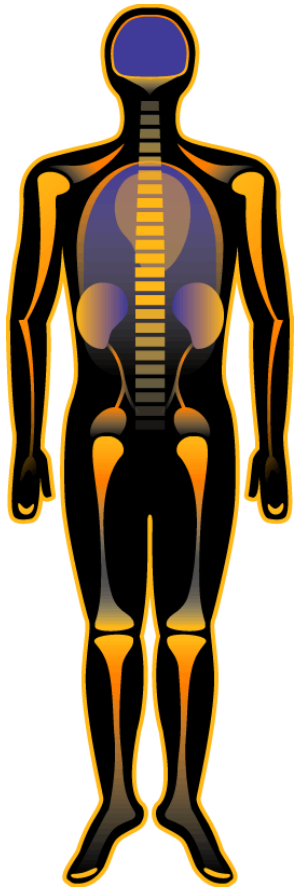
- **Autonomic Computing:** computing environment with the ability to manage itself and dynamically adapt to changes according to business policies and objectives.

# Autonomic option

- Autonomic computing
  - Named after autonomic nervous system
  - Systems can manage themselves according to an administrator's goals
  - Self-governing operation of the entire system, not just parts of it
  - New components integrate as effortlessly as a new cell establishes itself in the body
- First step
  - Examine the vision of autonomic computing



# Vision for Autonomic Computing



Intelligent systems that:

- Manage complexity
- Know themselves
- Continuously tune themselves
- Adapt to unpredictable conditions
- Prevent and recover from failures
- Provide a safe environment

We need self-managing computing systems

- Behavior specified by system administrators via high-level policies
- System and its components figure out how to carry out policies

# Self-management (1/2)

- Management
  - Changing components
  - External conditions
  - Hardware/software failures
- Component upgrade
  - Continually check for component upgrades
  - Download and install
  - Reconfigure itself
  - When it detects errors, revert to the older version

# Self-management (2/2)

- Four aspects of self-management
  - Self-configuration
    - Configure themselves automatically
    - High-level policies (what is desired, not how)
  - Self-optimization
    - Hundreds of tunable parameters
    - Continually seek ways to improve their operation
  - Self-healing
    - Fault-tolerance
    - Analyze information from log files and monitors
  - Self-protection
    - Malicious attacks
    - Cascading failures

# Autonomic Computing attributes

**Increased Flexibility**

**Adapt to dynamically changing environments**

**Business Resiliency**

**Discover, diagnose, and act to prevent disruptions**


**Operational Efficiency**

**Tune resources and balance workloads to maximize use of resources**

**Secure Information and Resources**

**Anticipate, detect, identify, and protect**

# Evolving towards Self-management

| | Today | The Autonomic Future |
|---|---|---|
| Self-configure | Corporate data centers are multi-vendor, multi-platform. Installing, configuring, integrating systems is time-consuming, error-prone. | Automated configuration of components, systems according to high-level policies; rest of system adjusts automatically. Seamless, like adding new cell to body or new individual to population. |
| Self-healing | Problem determination in large, complex systems can take a team of programmers weeks | Automated detection, diagnosis, and repair of localized software/hardware problems. |
| Self-optimize | Applications have hundreds of nonlinear tuning parameters; many new ones with each release. | Components and systems will continually seek opportunities to improve their own performance and efficiency. |
| Self-protect | Manual detection and recovery from attacks and cascading failures. | Automated defense against malicious attacks or cascading failures; use early warning to anticipate and prevent system-wide failures. |

# Architectural considerations (1/2)

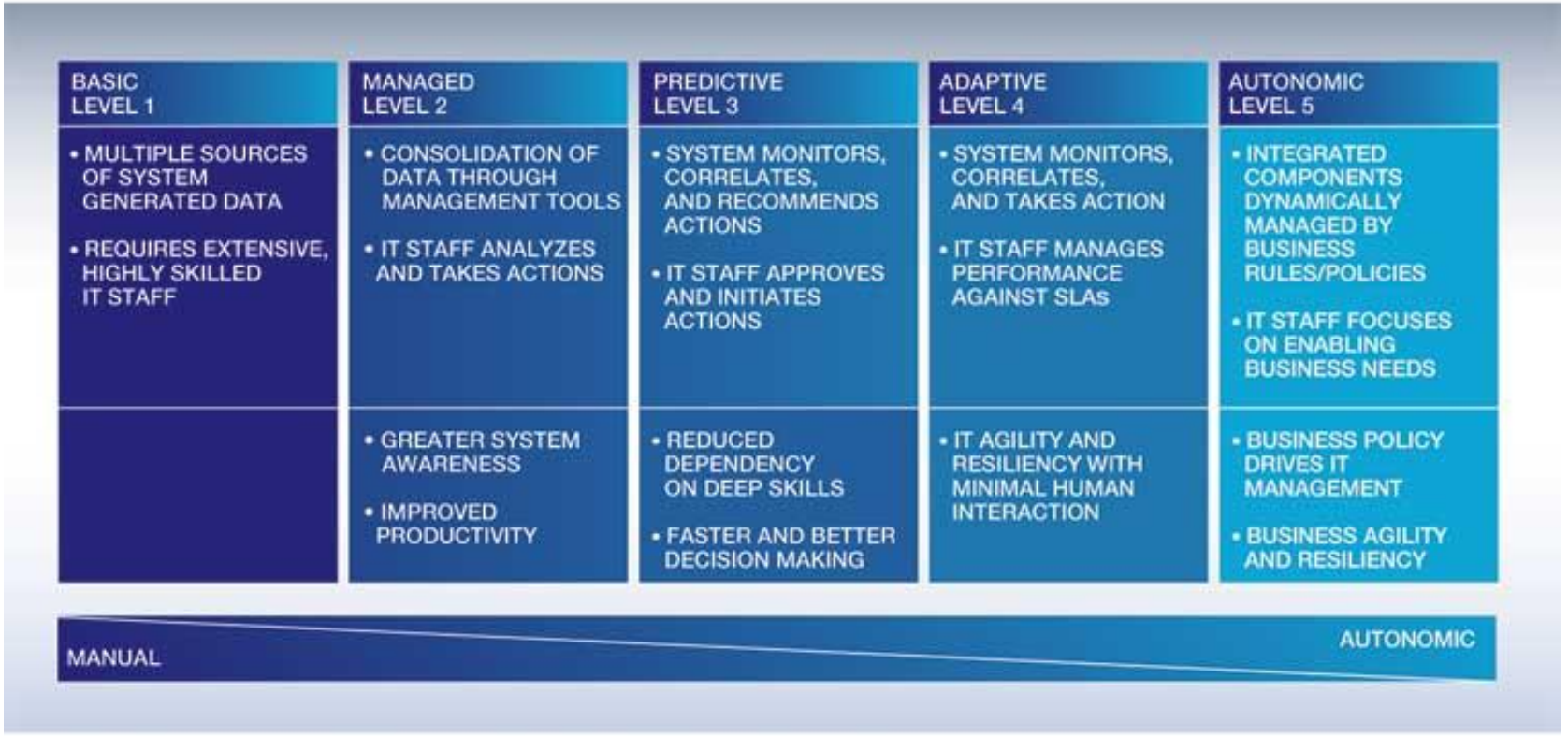
- Autonomic elements will manage
  - Internal behavior
  - Relationships with other autonomic elements
- An autonomic element consist of
  - Managed elements
    - Hardware/software resource
  - Autonomic manager
    - Monitoring the managed elements and external environment
- We can consider autonomic elements as software agents and autonomic systems as multi-agent systems

# Architectural considerations (2/2)

- Fully autonomic computing
  - Evolve as designers gradually add increasingly sophisticated autonomic managers to existing managed elements
- Autonomic elements will function at many levels
  - At the lower levels
    - Limited range of internal behaviors
    - Coded behaviors
  - At the higher levels
    - Increased dynamism and flexibility
    - Goal-oriented behaviors
- Relationships will evolve into flexible relationships that are established via negotiation

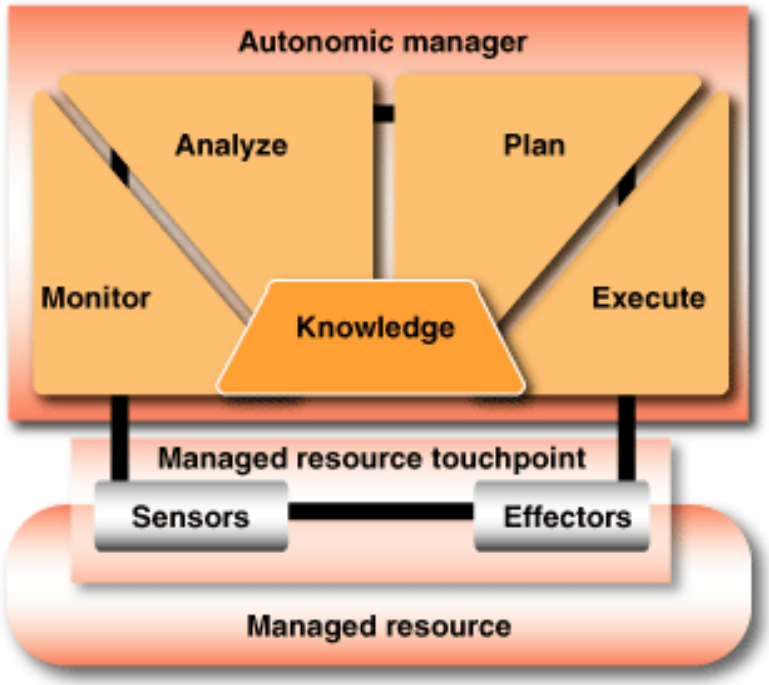
# Autonomic levels

Figure 2 Evolving to autonomic operations




From *IBM Global Services and Autonomic Computing*, IBM White Paper, October 2002; see http://www-3.ibm.com/autonomic/pdfs/wp-igs-autonomic.pdf.

# Autonomic Architecture



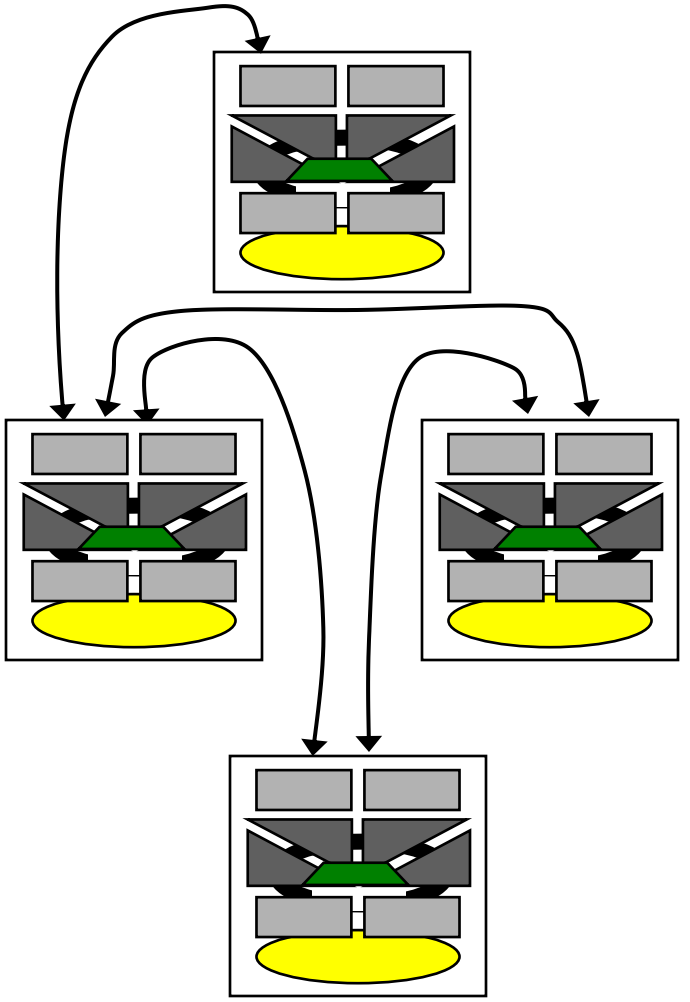

- An **autonomic manager** contains a continuous **control loop** (MAPE loop) that monitors activities and takes actions to adjust the system to meet objectives
- Autonomic managers learn from past experience to build action plans
- Managed elements need to be instrumented consistently

# Utility Functions

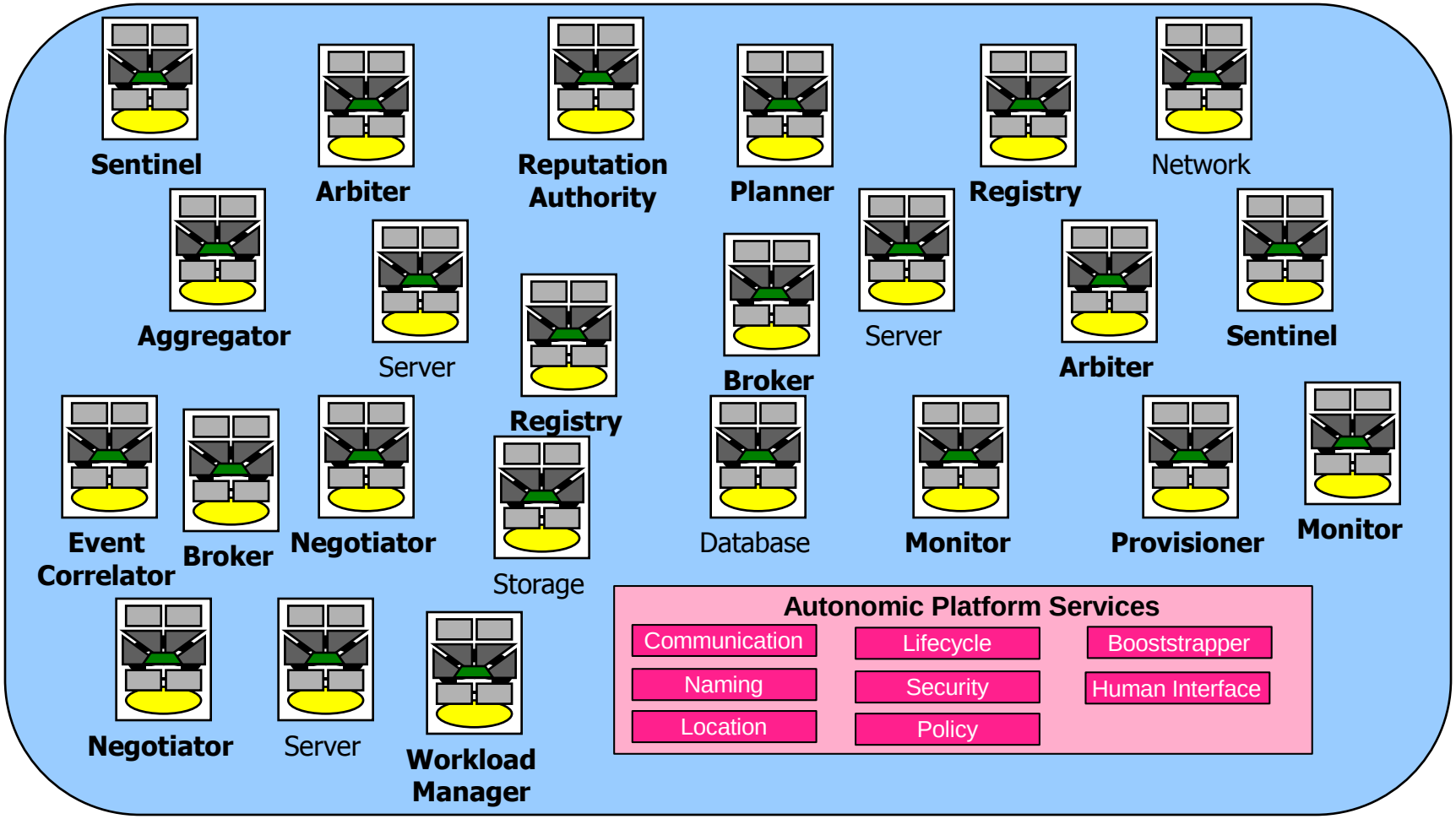
- Statements of preference described by humans.
- Function returning a single numerical value representing satisfaction, based on an input of variables.
- Utility functions are used for decision making, by assigning a value to specific service metrics, such as throughput and service time.
- These functions are used to achieve self management through high-level goal specification.
- Utility functions may be dynamically modified at run-time to ensure maximum utility

# Interaction

- Relationships
  - Dynamic, ephemeral
  - Formed by agreement
    - May be negotiated
  - Full spectrum
    - Peer-to-peer
    - Hierarchical
  - Subject to policies



# AC System and Infrastructure



# Open Standards for Self-Managing Systems

***Why Standards?***

- **Autonomic computing is an industry-wide initiative**
- **Proprietary solutions with vendor lock-in are unacceptable to clients**
- **Open, level playing field where vendors compete with best solutions**
- **Standards-based components can interoperate**
- **Easier to integrate multivendor components into an end-to-end solution**

***Plan…***

**Leverage existing standards when feasible**

**Drive new standards through open standards bodies when necessary**

**Coordinate disparate standards efforts when required**

# Standards

Heterogeneous IT environment needs standards:

- **WSDM (Web Services Distributed Management)**
  - Provide standard basis for management interfaces using web services
  - It can be realized in endpoint manageability interfaces, using sensor and effector interfaces
  - WSRF protocols were also used by WSDM as the means to interacts with manageable resources described in WSDM.
- **CIM-SPL** (Common Information Model-Simplified Policy Language)
  - Preliminary. Implemented in the Apache incubator project called Imperius

# Limitations

- Real-world systems take into account more service metrics than response time and number of servers. The system must be able to scale to handle many service metrics.
- Real-world applications also have more than two competing applications and one that simply waits until another is done with the resources.
- Real-world systems manage more than servers. The system must be able to handle service, data, and hardware resources simultaneously.
- Stability: When operating in an environment of frequent variation, the system may spend more time reacting and adjusting than processing workloads

# Engineering challenges (1/3)

- Life cycle of an autonomic element
  - Design, test, and verification
    - Testing autonomic elements will be challenging
  - Installation and configuration
    - Element registers itself in a directory service
  - Monitoring and problem determination
    - Elements will continually monitor themselves
    - Adaptation, optimization, reconfiguration
  - Upgrading
  - Uninstallation or replacement

# Engineering challenges (2/3)

- Relationships among autonomic elements
  - Specification
    - Set of output/input services of autonomic elements
    - Expressed in a standard format
    - Description syntax and semantics
  - Location
    - Find input services that autonomic element needs
  - Negotiation
  - Operation
    - Autonomic manager oversees the operation

# Engineering challenges (3/3)

- Systemwide issues
  - Authentication, encryption, signing
  - Autonomic elements can identify themselves
  - Autonomic system must be robust against insidious forms of attack
- Goal specification
  - Humans provide the goal and constraints
  - Indirect effect of policies
  - Ensure that goals are specified correctly in the first place
  - Autonomic systems will need to protect themselves from input goals that are inconsistent, implausible, dangerous, or unrealizable

# Scientific challenges

- Behavior abstractions and models
  - Mapping from local behavior to global behavior is a necessary
- Learning and optimization theory
  - Agents continually adapt to their environment that consists of other agents
  - There are no guarantees of convergence
- Negotiation theory
- …

# References


- The Vision of Autonomic Computing. Jeff Kephart, David M. Chess. IBM Research
- Applications of Multi-Agent Learning in E-Commerce and Autonomic Computing. Jeff Kephart. IBM Research
- IBM Autonomic Computing. Kamran Saleem Soomro, Ammar Lodhi. NTNU