ACRA
AC Reference Architecture

- An Autonomic System (AS) consists of a collection of Autonomic Elements
- An Autonomic Element (AE)
  - Contains resources and delivers services to humans or other autonomic elements
  - Manages its behaviour in accordance with policies that humans or other AEs have established
  - Acts like an agent
    - Autonomous, proactive, goal-directed
    - Interacts with environment

Kephart, Chess: IEEE Computer, 36(1):41-50, Jan. 2003

Autonomic Architecture Blueprint

- System building blocks or components
- External interfaces and behaviors of individual building blocks
- Composition and interactions of building blocks so that the individual blocks can contribute toward a common goal
- Composition of blocks into systems so that the the system as a whole is self-managing
- Data and control integration standards

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

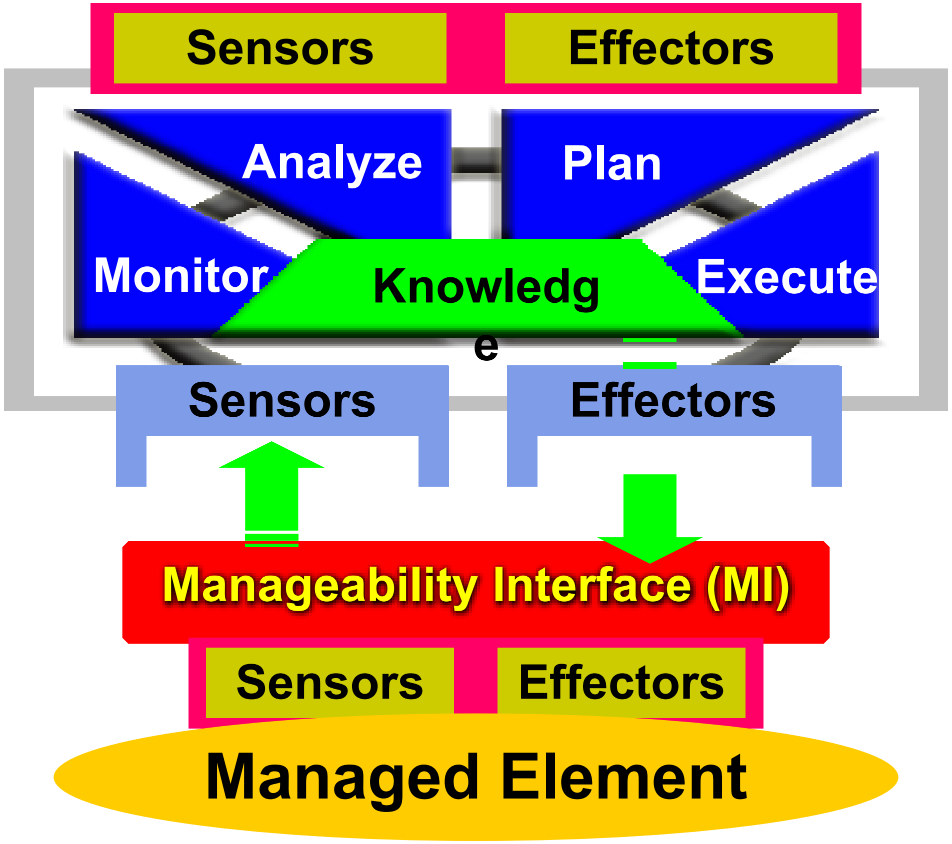
Architectural Building Blocks

- Autonomic System (AS)

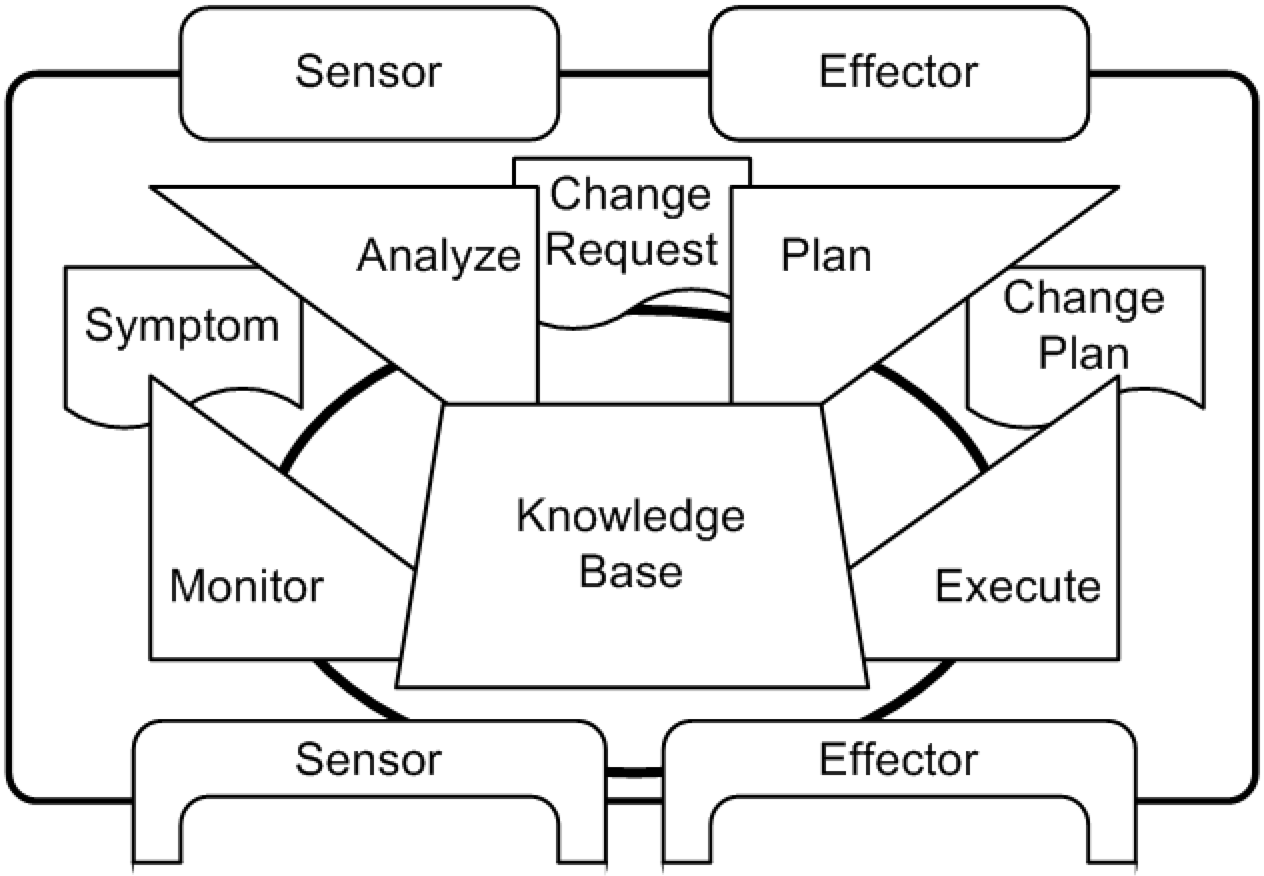
- Autonomic Element (AE)
- Autonomic Manager (AM)
- Managed Element (ME)

- Manageability Endpoint (ME)
- Manageability Interface (MI)

- Knowledge sources
- Enterprise service bus

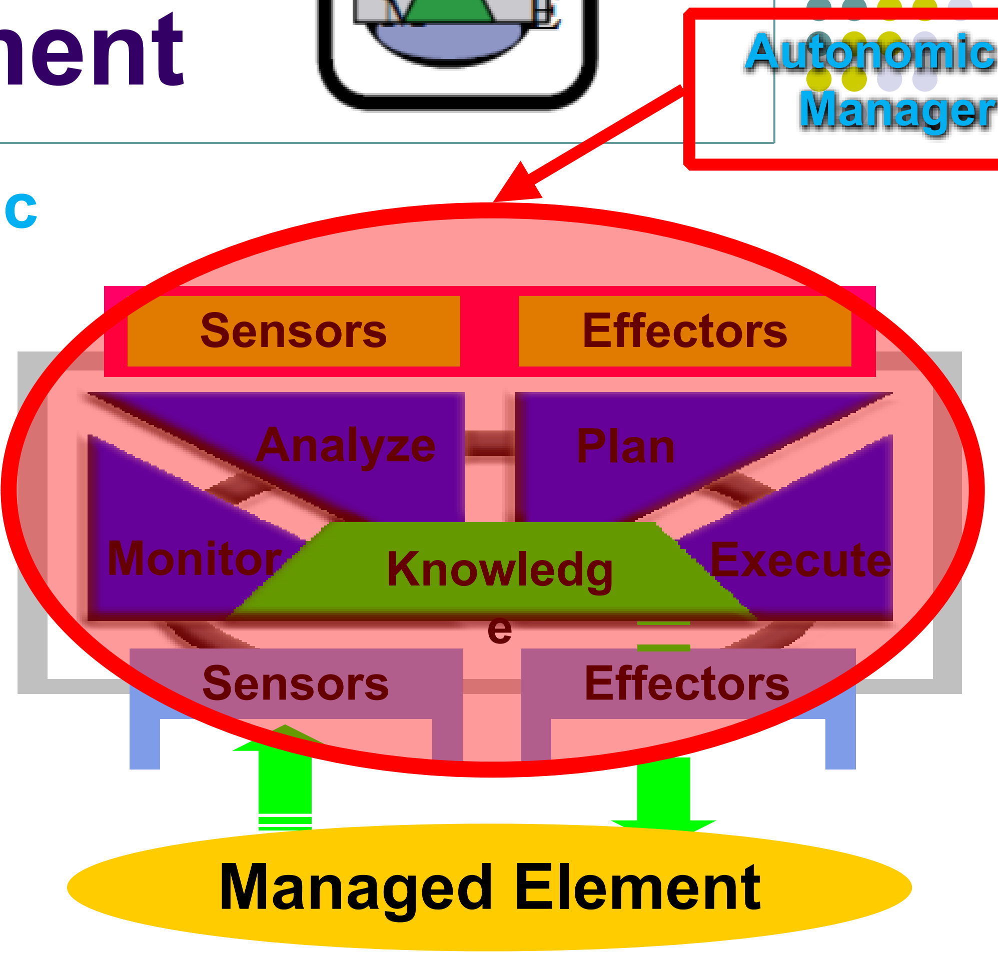


Autonomic Manager



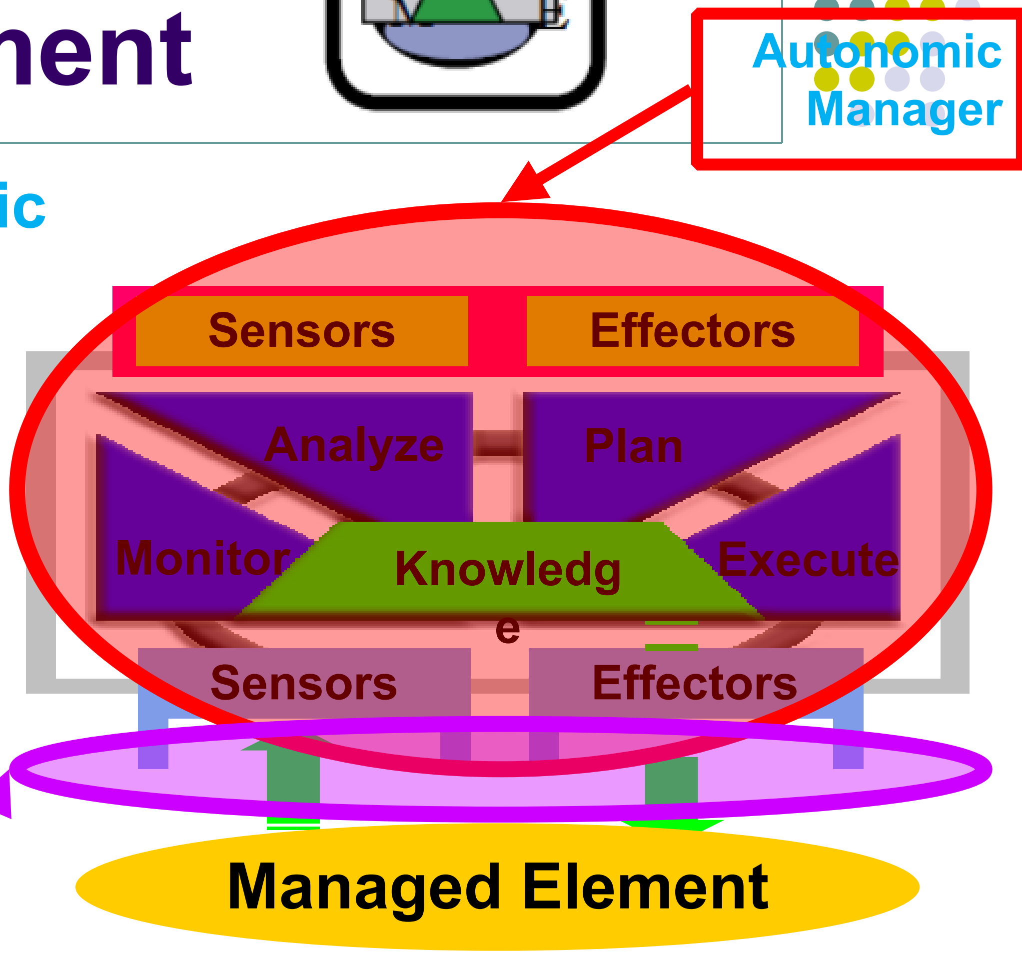
Autonomic Element

- Consists of an **Autonomic Manager (AM)** and an Autonomic Element (AE)



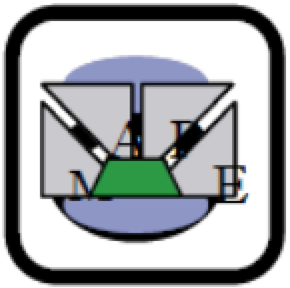
Autonomic Element

- Consists of an **Autonomic Manager (AM)** and an Autonomic Element (AE)
- Manager and managed element form a **level of indirection**
  - Spatially and temporally separate entities
  - Enterprise Service Bus



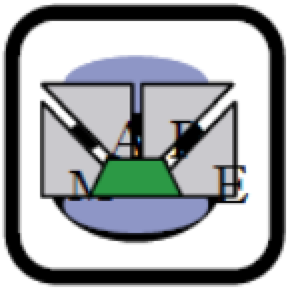
Level of indirection

Autonomic Manager



- An Autonomic Manager (AM) automates some management function and externalizes its behavior in its **Manageability Interface (MI)**
- An AM implements an **intelligent control loop**
- **Self-management—an automated method to**
  - Collect information from a resource
  - Filter information and store in a repository
  - Analyze information
  - Create a plan as a sequence of actions
  - Execute those actions

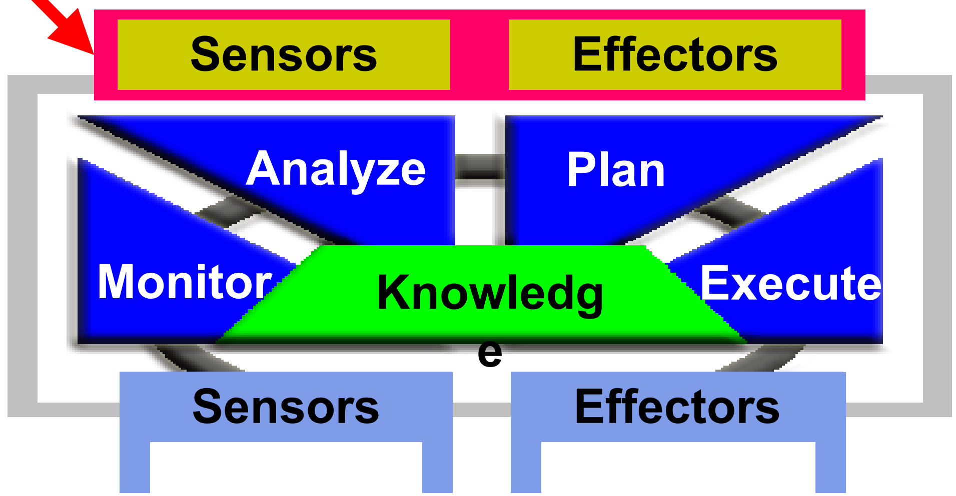
Autonomic Manager



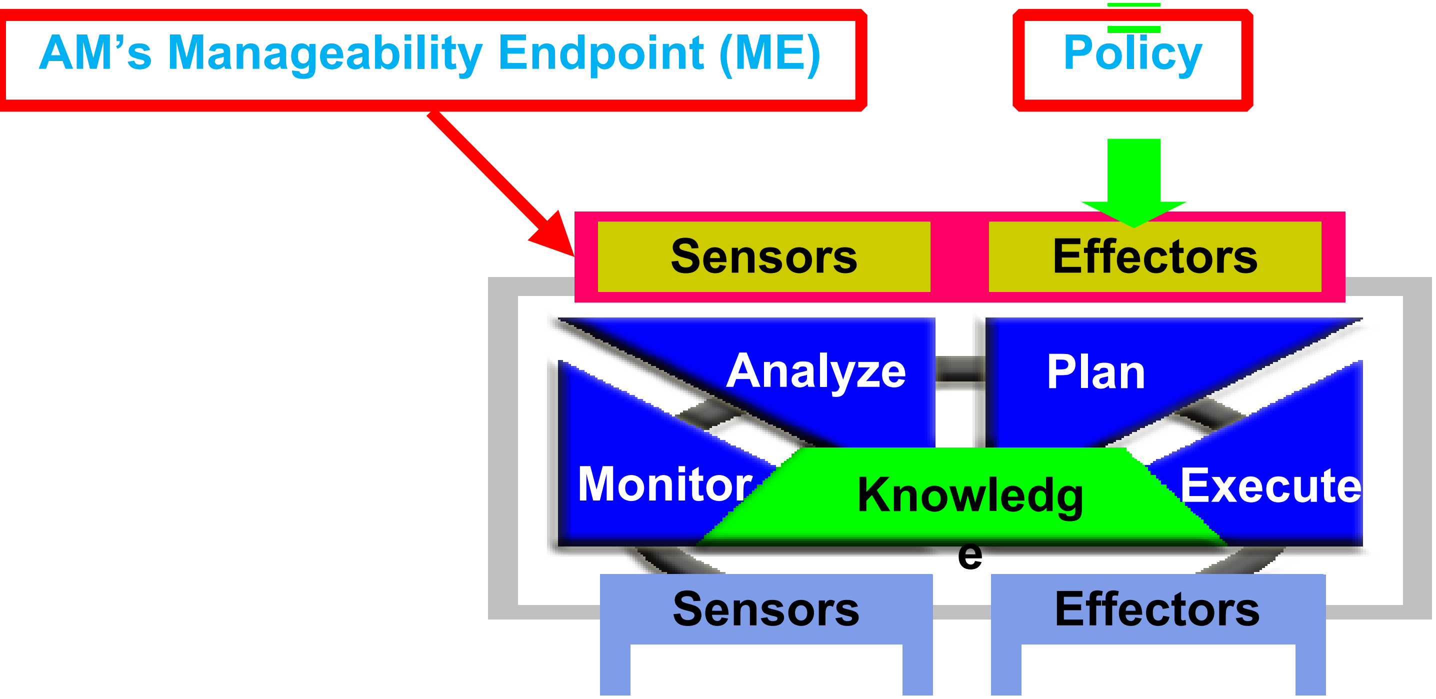
- Manages a resource
  - Storage
  - CPU
  - Database
  - Service
  - Legacy system
  - Another Autonomic Element
- Interacts with the managed element through
  - Sensors and effectors
  - Manageability interface
  - Manageability endpoint

Autonomic Manager

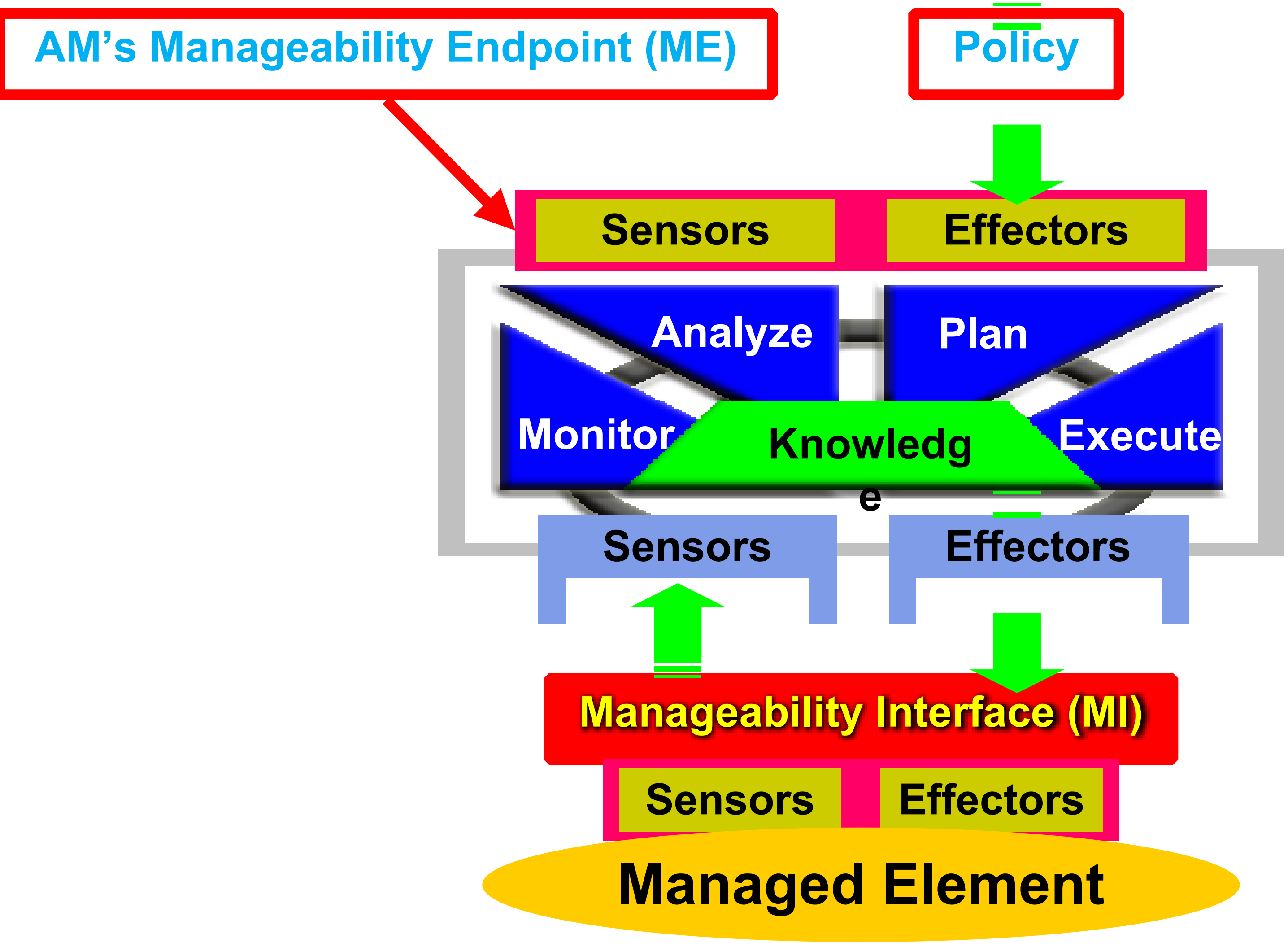
AM’s Manageability Endpoint (ME)



Autonomic Manager

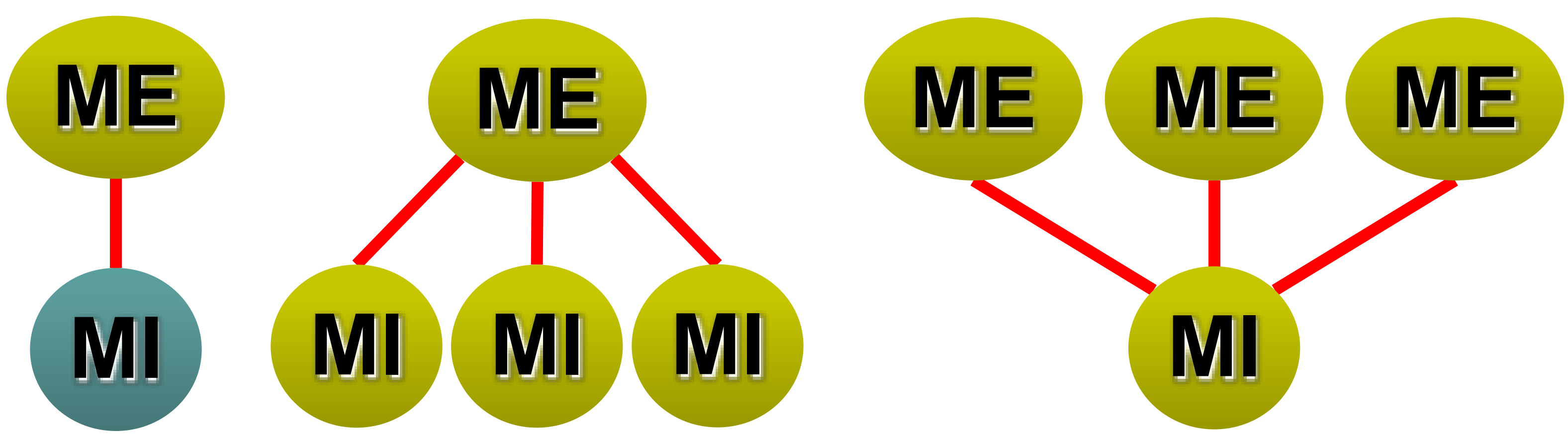


Autonomic Manager



Manageability Endpoints

- A Manageability Endpoint (ME) exposes the state and the management operations for a resource
- An autonomic manager communicates with a manageability endpoint through the Manageability Interface (MI)

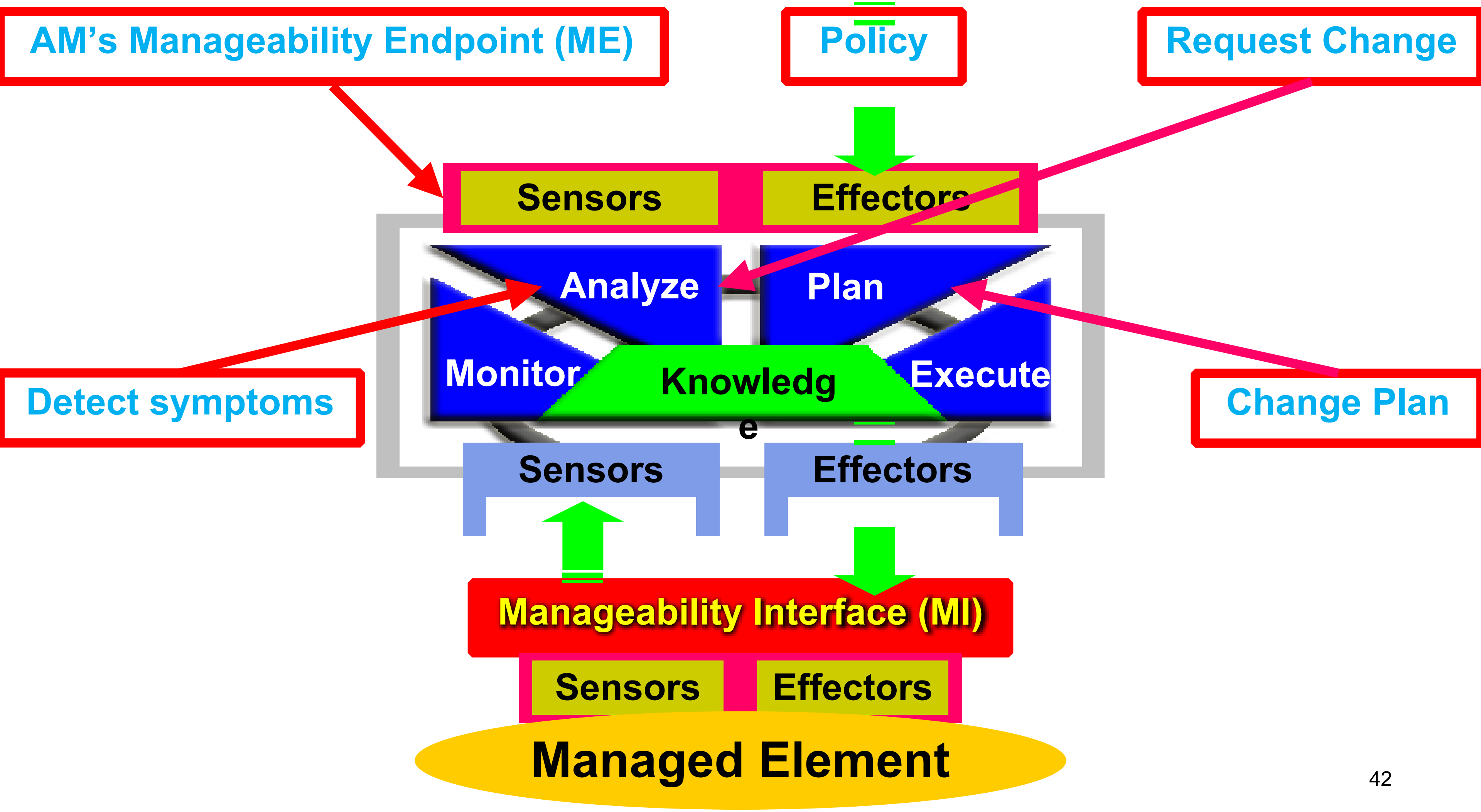


Manageability Interface

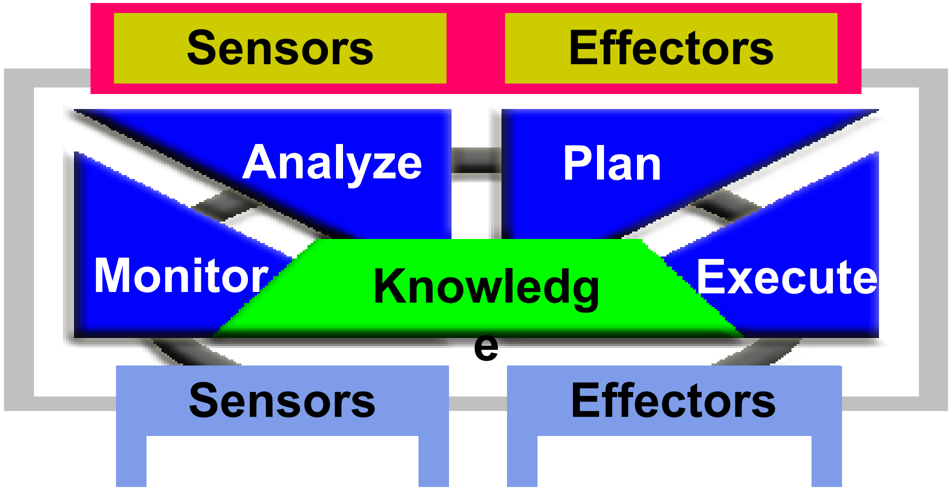
- An MI for monitoring and controlling a managed resource consists of sensors and effectors
- Sensors obtain data from the resource
  - read state variables in the ME
- Effectors perform operations on the resource
  - call methods in the ME

- Critical success factors for AC initiative
  - Separating AMs and MEs
  - Standardizing MIs

Autonomic Manager



Autonomic Manager
MAPE Loop or MAPE-K Loop



MAPE-K Loop

Monitor

- Senses the managed process and its context
- Collects data from the managed resource
- Provides mechanisms to aggregate and filter incoming data stream
- Stores relevant and critical data in the knowledge base or repository for future reference.

Analyzer

- Compares event data against patterns in the knowledge base to diagnose symptoms and stores the symptoms
- Correlates incoming data with historical data and policies stored in repository
- Analyzes symptoms
- Predicts problems

MAPE-K Loop Planner

- Interprets the symptoms and devises a plan
- Decides on a plan of action
- Constructs actions
  - building scripts
- Implements policies
- Often performed manually

Execute Engine

- Executes the change in the managed process through the effectors
- Perform the execution plan
- Often performed manually

MAPE-K Loop
Knowledge Base

- The four components of a MAPE-K loop work together by exchanging knowledge through the **knowledge base** to achieve the control objective.
- An autonomic manager
  - maintains its own knowledge
    - Information about its current state as well as past states
  - But also has access to knowledge which is shared among collaborating autonomic managers
    - Configuration database, symptoms database, business rules, provisioning policies, or problem determination expertise

Design Considerations
Monitor

- The monitor function provides the mechanisms that collect, aggregate, filter and report details collected from a managed resource.
  - What kinds of data and events are collected from which sources, sensors, or probes?
  - Are there common event formats?
  - What is the sampling rate and is it fixed or varying?
  - Are the sampled sources fixed or do they change dynamically?
  - What are appropriate filters for the data streams?

Design Considerations
Monitor

- A large portion of the knowledge base is monitored information
  - How much information is needed for future reference?
  - With the detailed level of reporting and logging functions within software systems today, it is important to monitor and store data that is really going to be of use to the control loop
  - If large amounts of log data, for example, are stored, performance might deteriorate because data is constantly being monitored when it has no relevance to the system.

Design Considerations
Analyzer

- The analyzer provides mechanisms to correlate and model complex situations
  - Embodies the control model together with the planner
  - Time-series forecasting and queuing models
- These mechanisms allow the autonomic manager to learn about the IT environment and help predict future situations.
  - How are the collected data represented and stored?
  - What are appropriate diagnosis methods to analyze the data?
  - How is the current state of the system assessed?
  - How much past state needs to be kept around?
  - How are critical states archived?
  - How are common symptoms recognized (e.g., symptoms db)?

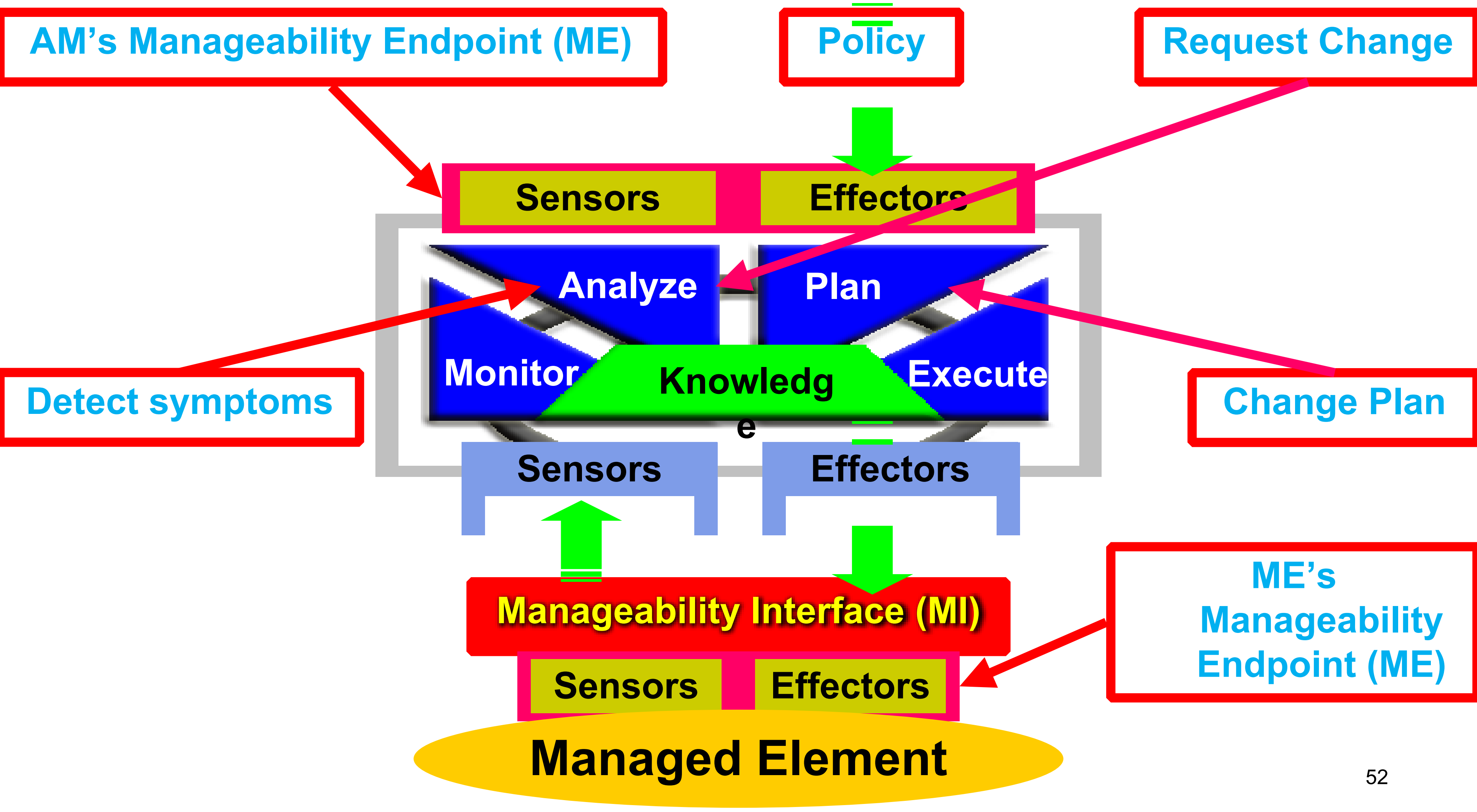
Design Considerations
Planning Engine

- The planning engine provides mechanisms to construct the actions needed to achieve goals and objectives.
- It uses policy information to guide its work.
  - How is the future state of the system inferred and how is a decision reached?
    - With off-line simulation, quality of service (QoS) objectives, or utility goal functions
  - What models and algorithms are used for trade-off analysis?
  - What are the priorities for adaptation across multiple control loops and within a single control loop?
  - Under what conditions should adaptation be performed?
    - Allow for head-room or avoid system thrashing while considering timing issues relating to the required adaptations

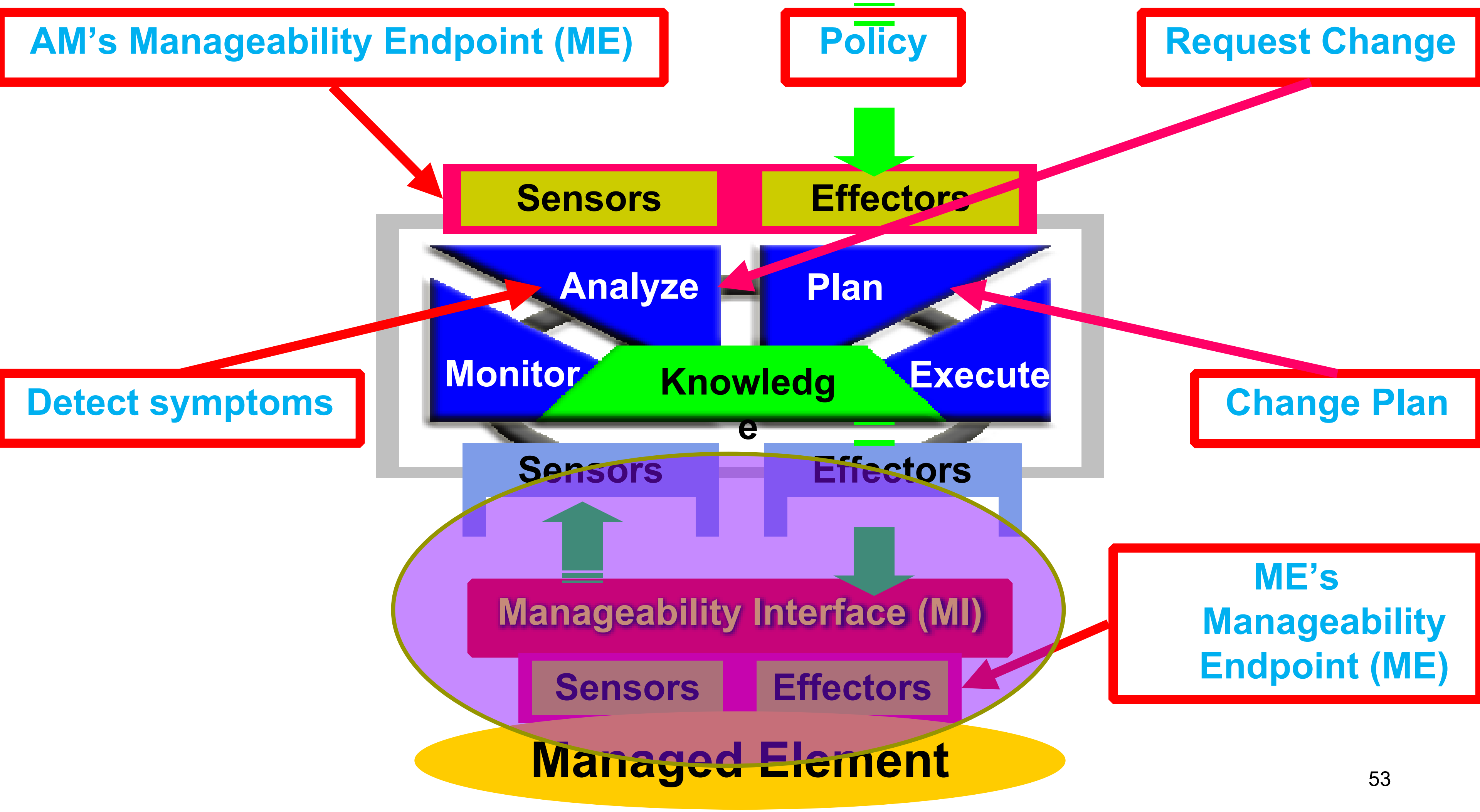
Design Considerations
Execution Engine

- The execution engine provides mechanisms to control the execution of a plan by updating the managed element dynamically.
  - What are the managed elements and how can they be manipulated?
    - By parameter tuning
    - By injecting new algorithms
  - Are changes of the system pre-computed opportunistically assembled, composed, or generated?
    - Switching between known configurations

Autonomic Manager



Autonomic Manager



Autonomic Element Architecture

	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