

Engineering

Self-Adaptive Systems

Report from Dagstuhl 08031

Yuriy Brun, Giovanna Di Marzo Serugendo
Cristina Gacek, Holger Giese, Holger Kienle
Marin Litiou, Hausi Müller, Mary Shaw

SEAMS 2008 Workshop
Leipzig, May 12-13, 2008

Engineering Self-Adaptive Systems

Report from Dagstuhl 08031

High degree of dynamism

Goals or policies changed at run-time

V&V performed at run-time

High Expectations for Self-Adaptability

Software systems must become more

- ❖ versatile, flexible, resilient, dependable
- ❖ robust, energy-efficient, recoverable
- ❖ customizable, configurable
- ❖ self-optimizing

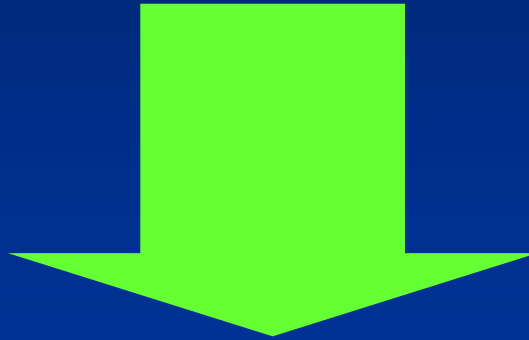
by adapting to changing operational
contexts and environments

Typical System Properties Accommodated by Self-Adaptive Systems

- Decentralized operation and control
- Conflicting, unknowable, diverse requirements
- Continuous evolution and deployment
- Heterogeneous, inconsistent, changing elements
- Indistinct people/system boundary
- Normal failures
- New forms of acquisition and policy

Managing Tradeoffs

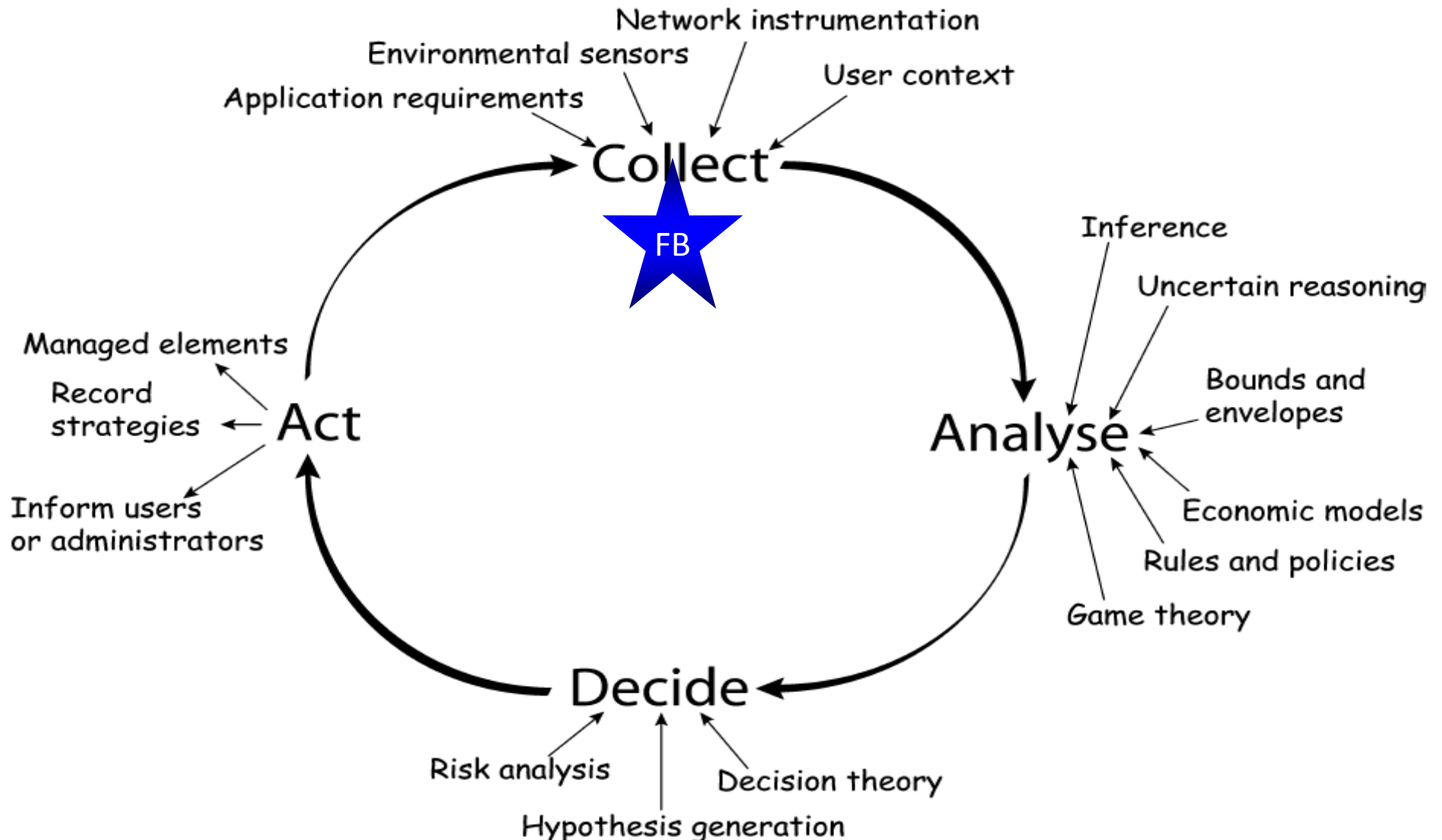
- **From** satisfaction of requirements through traditional, top-down engineering



- **To** satisfaction of requirements by regulation of complex, decentralized systems

*How much environment uncertainty can we afford?
What benefits do we accrue by accommodating context uncertainty?*

Approach Feedback Loop



Self-Adaptive versus Self-Organizing

Self-adaptive systems

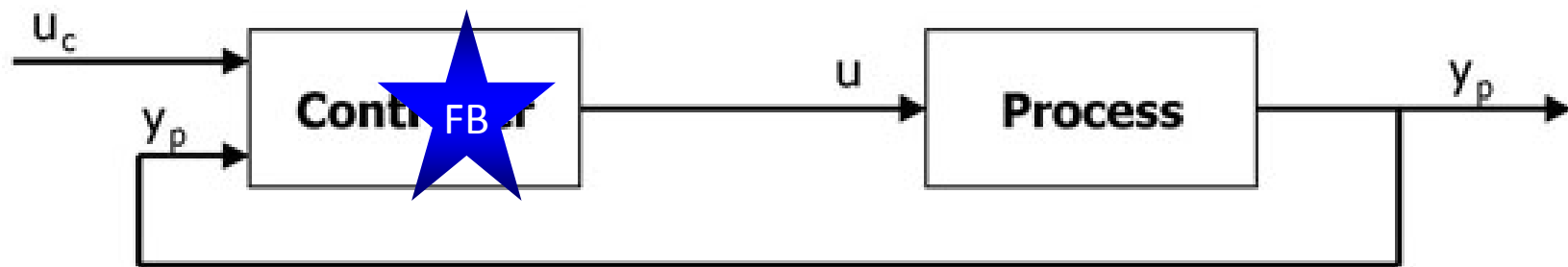
- Work in a top-down manner
- Analyze changing environment
- Selecting among available adaptations
- Explicit internal representation of themselves and their global goals

Self-organizing systems

- Work in a bottom-up manner
- Employ a cooperative approach
- Made of simple components that follow simple rules
- Components are not aware of their environment or their adaptation options
- Out of those rules emerges desired system behavior

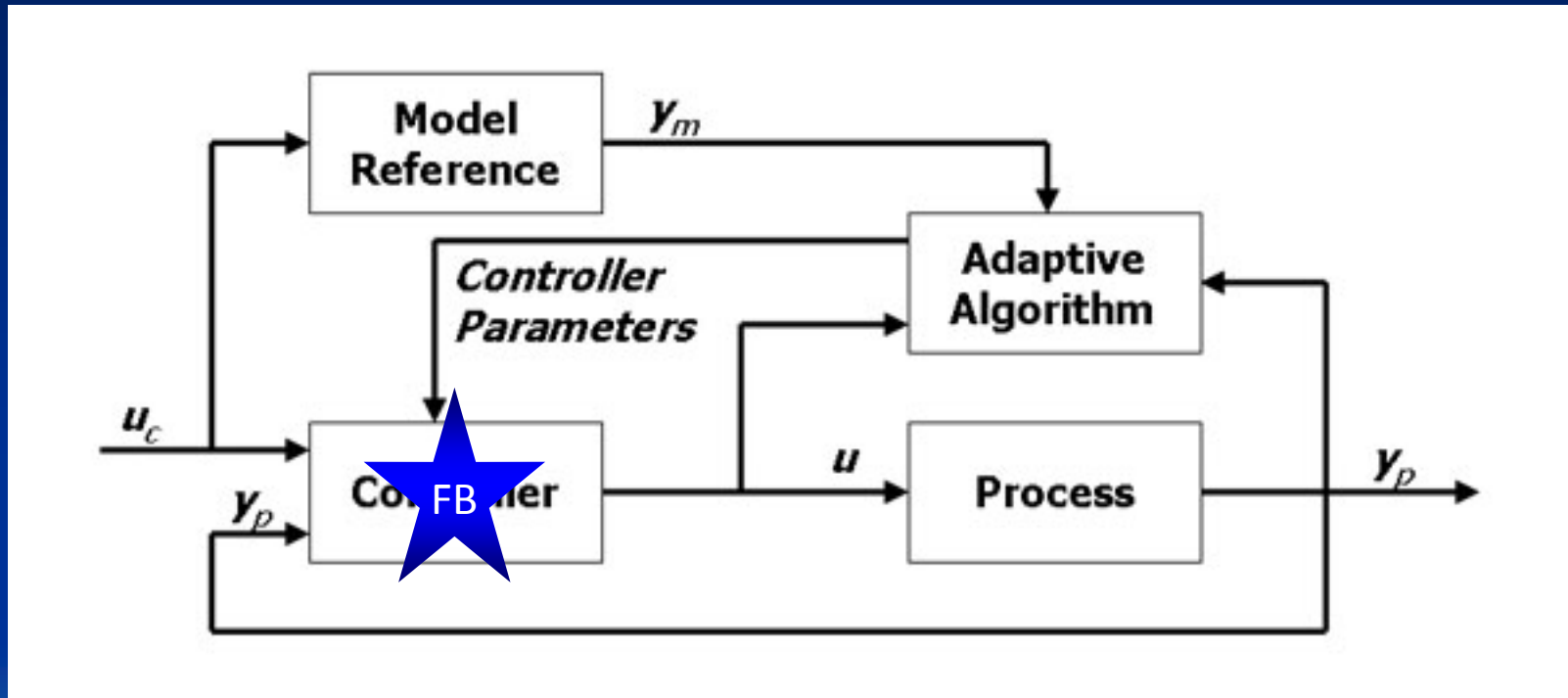
Software ecosystems incorporate both

Simple Control Loop



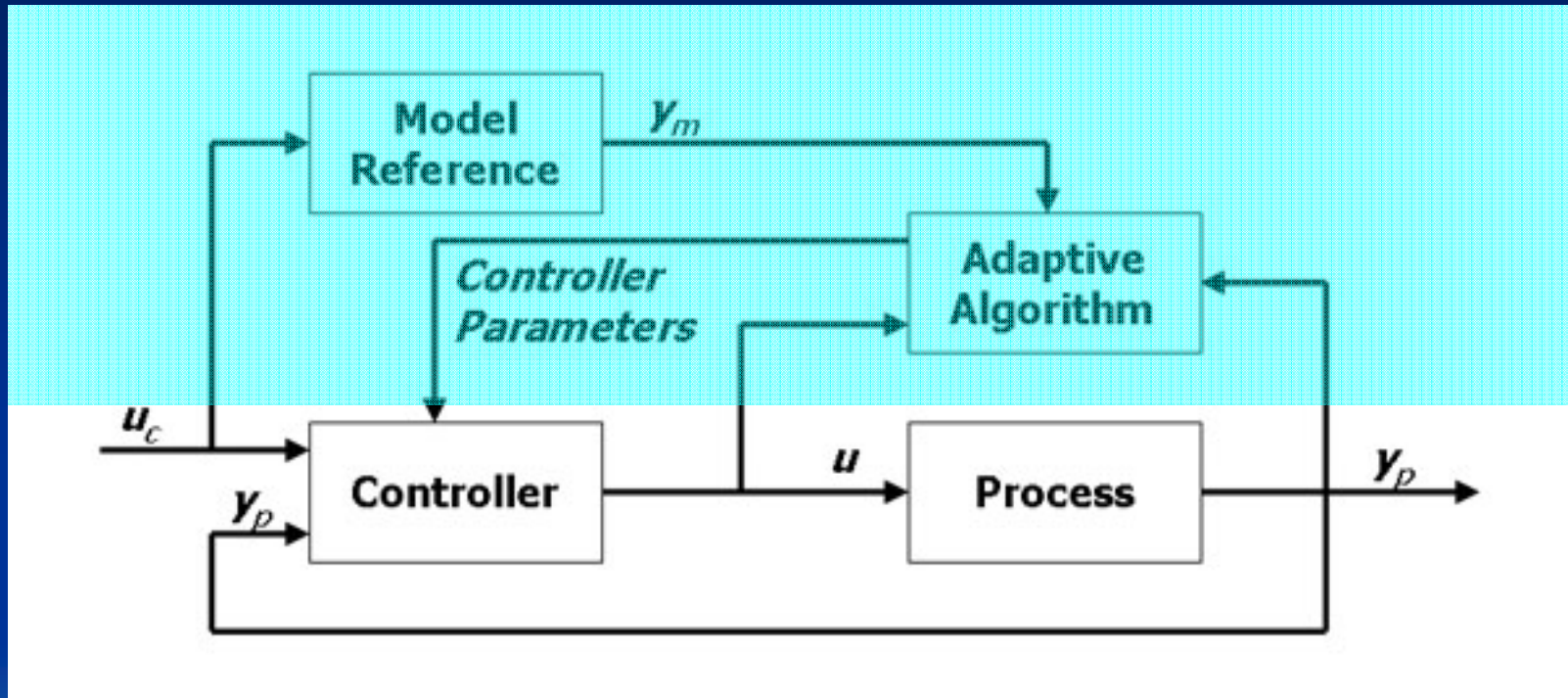
MRAC

Model-Reference Adaptive Control



MRAC

Model-Reference Adaptive Control



Software engineers know how to model systems

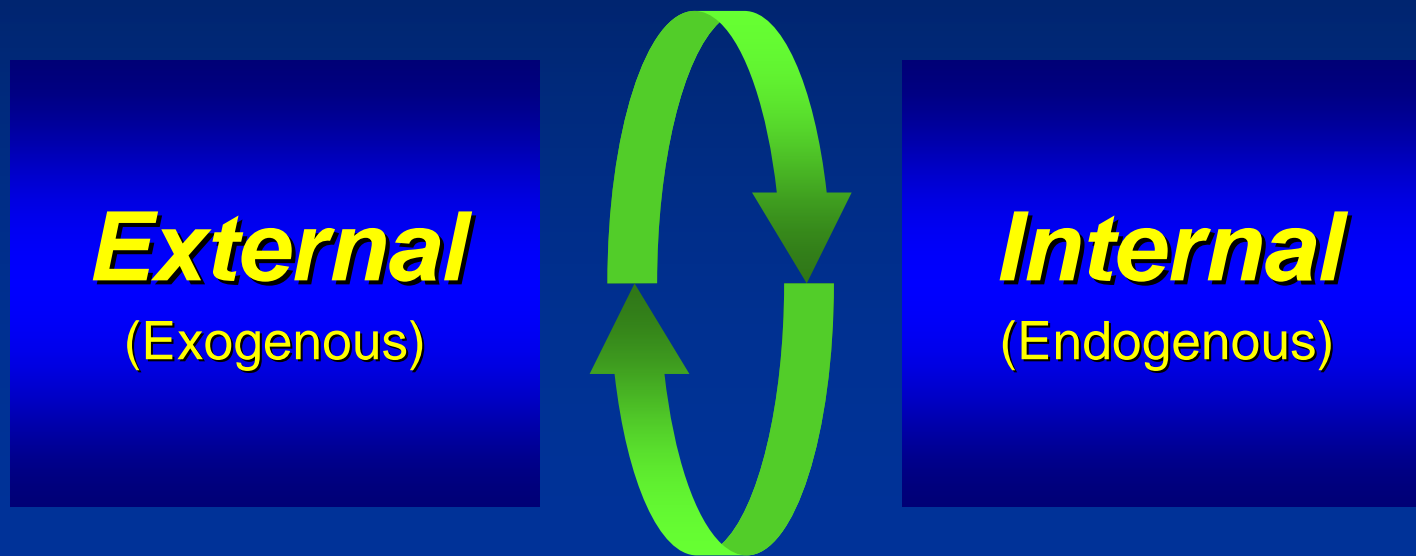
Multiple Control Loops

- Systems often embody multiple control loops
- Good engineering practice calls for making them independent whenever possible
- Sometimes they can operate independently, and sometimes they form a coherent hierarchy, but other types of interaction are possible too
- When multiple control loops interact, system design and analysis must cover their interactions

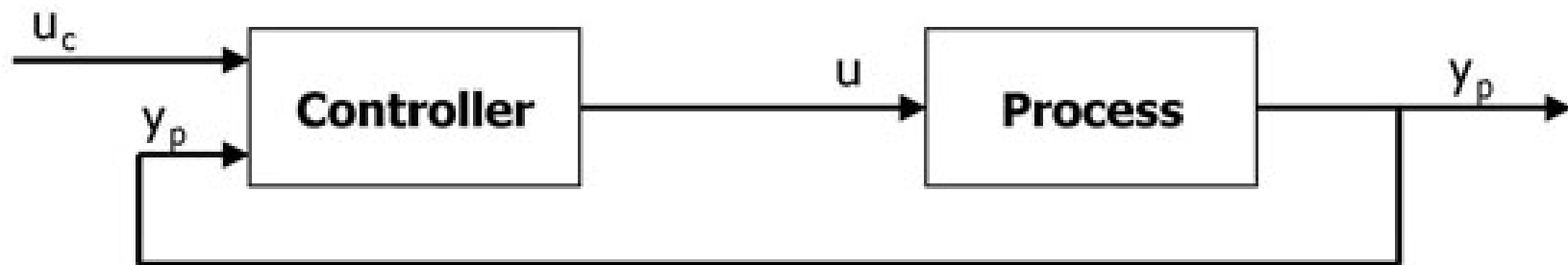
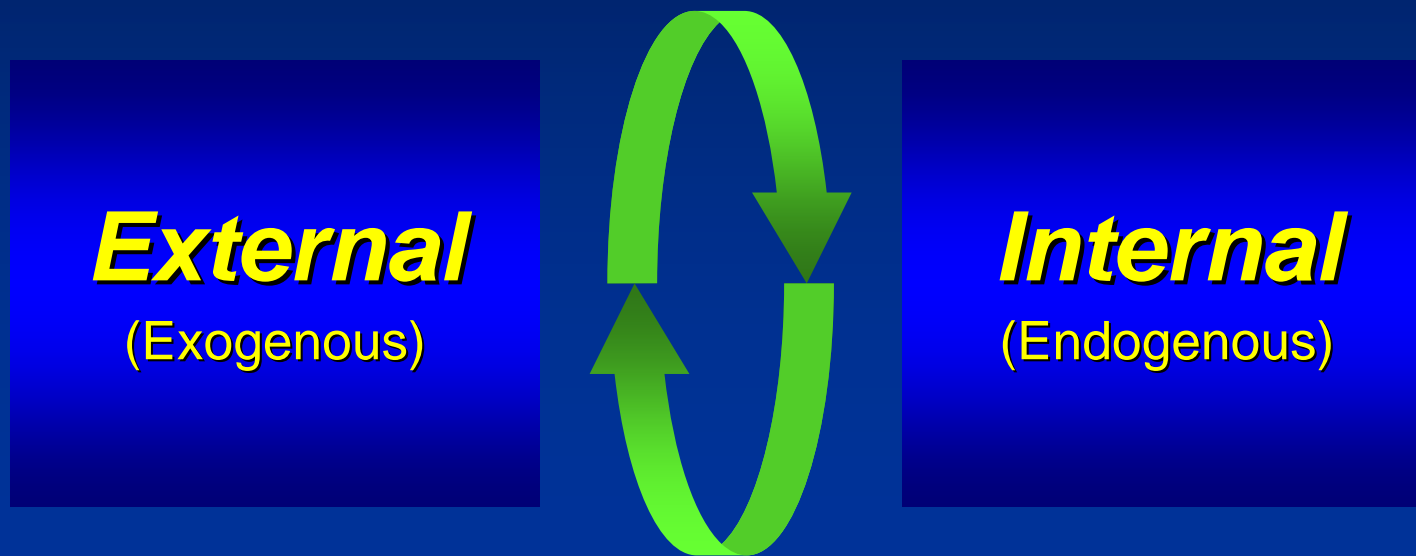
Arrangements of Control Loops



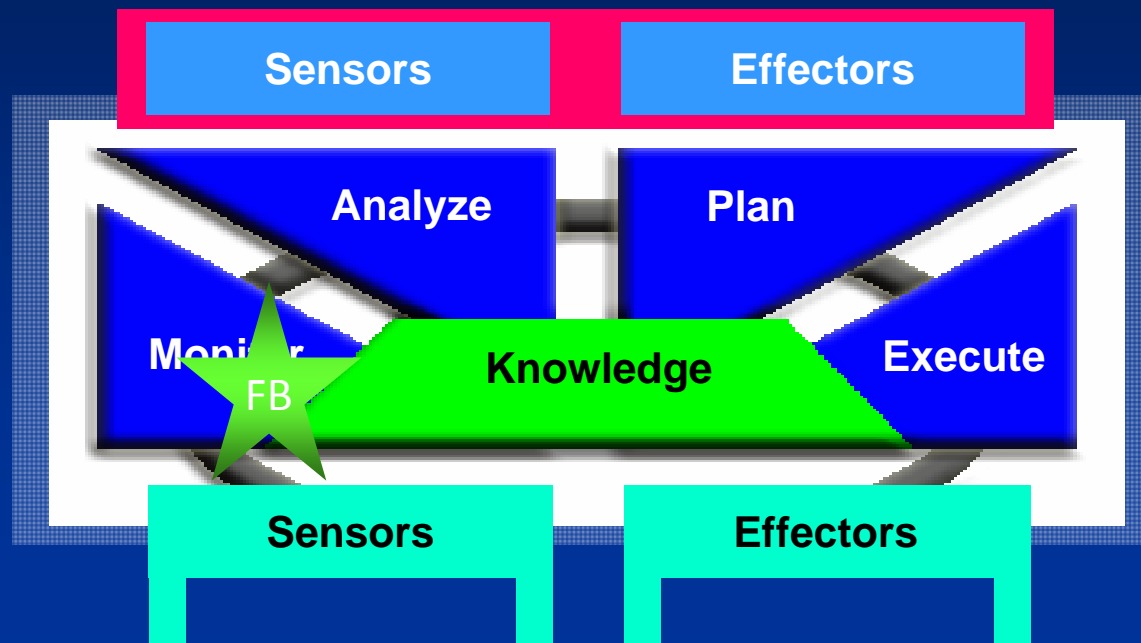
External versus Internal Self-Adaptation



External versus Internal Self-Adaptation



Autonomic Element



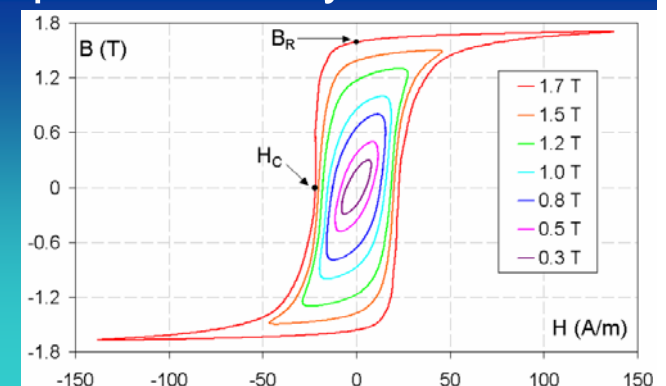
**IBM: An Architectural Blueprint for Autonomic Computing
White Paper, 4th Edition, June 2006**

Obligations for Control Loops

- Probes
 - ❖ What gets probed (i.e., which variables)?
 - ❖ Accuracy, authenticity, sensitivity of probes
 - ❖ Sampling rate, timing, variable, constant
- Monitor
 - ❖ Level of monitoring
 - ❖ Degree of filtering, cleaning in monitoring?
- Analysis engine
 - ❖ Degree of analysis, reasoning
- Planning/execution engine
 - ❖ Command authority the controller has over the process
- Actuators
 - ❖ Accuracy, authenticity, sensitivity of actuators
 - ❖ Sampling rate, timing, variable, constant

Obligations for Control Loops

- Knowledge base
 - ❖ How much self knowledge or awareness does the control loop have?
 - ❖ How much knowledge is available from the context or the environment?
 - ❖ How much history is retained?
 - ❖ How much prediction is achieved?
- Degree of automation in the controller
- Support for hysteresis
 - ❖ any number of states, independent of the inputs to the system
 - ❖ Control trashing
 - ❖ Well-being region
 - ❖ Dynamically abstracting state space



Engineering Challenges

- Modeling
 - ❖ Models for MRAC
 - ❖ Mine experiences from control theory and various engineering fields
 - ❖ Utility functions
- Architecture
 - ❖ Vocabulary of control loop practice
 - ❖ Arrangements of control loops
 - ❖ Patterns for various problems and applications
 - ❖ Mechanisms
- Design
 - ❖ Abstracted vs. hidden control loops
 - ❖ Predictive, prognostics
 - ❖ Component and system level support
 - ❖ How much uncertainty can we afford in the environment?
 - ❖ What benefits do we accrue from accommodating uncertainty?
 - ❖ Intensive instrumentation, build in control loops from the ground up
- V&V
 - ❖ Run-time support
 - ❖ Reflection for requirements