Keywords: Autonomic Computing, Self-Managing, Self-Healing, Self-Configuring, Self-Protecting, Self-Optimizing

Abstract: Autonomic Computing is a novel initiative adopted by industries and organizations world-wide which aims to help address complexity is today's IT systems. The term is derived from human biology, where the autonomic nervous system monitors corporal stimuli without the need to contact the central nervous system. Likewise, it's the goal of Autonomic Computing to breed novel software solutions that can function in an autonomic way, without the need to depend upon complex and centralized management software and without the need of a human operator to take decisions. The Autonomic Computing architecture lays out a roadmap for the implementation of true Self-Managing software systems.

Agenda

- Introduction
- The Autonomic Computing architecture
  - Autonomic Computing vision
  - Autonomic Computing levels of maturity
  - Autonomic Computing disciplines
  - Autonomic Computing building blocks
  - Autonomic Computing mechanisms
- Knowledge formats and processing
- Roadmap to Self-Managing business processes
- Conclusion

4th Latin American Network Operations and Management Symposium (LANOMS 2005)

Introduction – concepts & landscape

Architecture – terms & ideas

Knowledge – internals of an autonomic manager

Roadmap – business processes and the automation goal

Conclusion – enablement and incentives for novel solutions

Introduction

- The Autonomic Computing initiative is an industry-wide, standards-based, open and largely adopted initiative.
- The goal is to provide a framework for interoperation of self-managing systems.
- Will be largely adopted and pervasive within the industry.
- Builds upon well known and proven technologies: policy management, data mining, event correlation, resource models, etc.

4th Latin American Network Operations and Management Symposium (LANOMS 2005)

The Autonomic Computing initiative is backed up by more than 60 industries worldwide. IBM, Fujitsu, Sun, HP, Microsoft all have large initiatives that are geared towards autonomic computing. Some have different names (like Sun’s N1 and Fujitsu’s Organic-IT) but they all support the same ideologies. Likewise, several research organizations and Universities around the globe work in projects related to Autonomic Computing: Berkeley, Duke, Cornell, U of Arizona, U of Bologna, Italia, Rutgers, NCSU, etc.

Autonomic Computing assets are being driven as open standards. See for example Project Hyades and standard activities in the OASIS group, DMTF, IETF, Java Community, etc:

CBE – Common Base Event

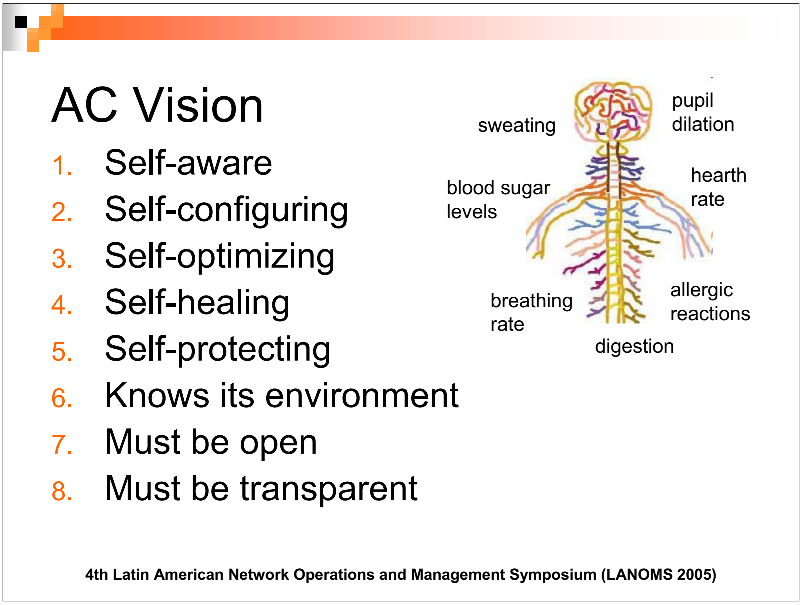
WEF – WS-DM Event Format

CIM – Common Information Model

RFC3060 – Policy Core Information Model

JSR87 – Java Agent Services

Etc



1. An autonomic computing system needs to "know itself" - its components must also possess a system identity. Since a "system" can exist at many levels, an autonomic system will need detailed knowledge of its components, current status, ultimate capacity, and all connections to other systems to govern itself. It will need to know the extent of its "owned" resources, those it can borrow or lend, and those that can be shared or should be isolated.
2. An autonomic computing system must configure and reconfigure itself under varying (and in the future, even unpredictable) conditions. System configuration or "setup" must occur automatically, as well as dynamic adjustments to that configuration to best handle changing environments.
3. An autonomic computing system never settles for the status quo - it always looks for ways to optimize its workings. It will monitor its constituent parts and fine-tune workflow to achieve predetermined system goals.
4. An autonomic computing system must perform something akin to healing - it must be able to recover from routine and extraordinary events that might cause some of its parts to malfunction. It must be able to discover problems or potential problems, then find an alternate way of using resources or reconfiguring the system to keep functioning smoothly.
5. A virtual world is no less dangerous than the physical one, so an autonomic computing system must be an expert in self-protection. It must detect, identify and protect itself against various types of attacks to maintain overall system security and integrity.
6. An autonomic computing system must know its environment and the context surrounding its activity, and act accordingly. It will find and generate rules for how best to interact with neighboring systems. It will tap available resources, even negotiate the use by other systems of its underutilized elements, changing both itself and its environment in the process - in a word, adapting.
7. An autonomic computing system cannot exist in a hermetic environment. While independent in its ability to manage itself, it must function in a heterogeneous world and implement open standards -- in other words, an autonomic computing system cannot, by definition, be a proprietary solution.
8. An autonomic computing system will anticipate the optimized resources needed while keeping its complexity hidden. It must marshal IT resources to shrink the gap between the business or personal goals of the user, and the IT implementation necessary to achieve those goals - without involving the user in that implementation.

AC Levels of Maturity

BASIC	MANAGED	PREDICTIVE	ADAPTIVE	AUTONOMIC
Requires high skilled IT staff	IT staff analyzes and takes actions	IT staff approves and initiates actions	IT staff manages the performance of the system	IT staff focuses on enabling the business
Manual tasks and processes	Greater system awareness	Faster and better decision making	Agility and resiliency with minimal human interaction	Business agility and resiliency

MANUAL → **AUTONOMIC**

4th Latin American Network Operations and Management Symposium (LANOMS 2005)

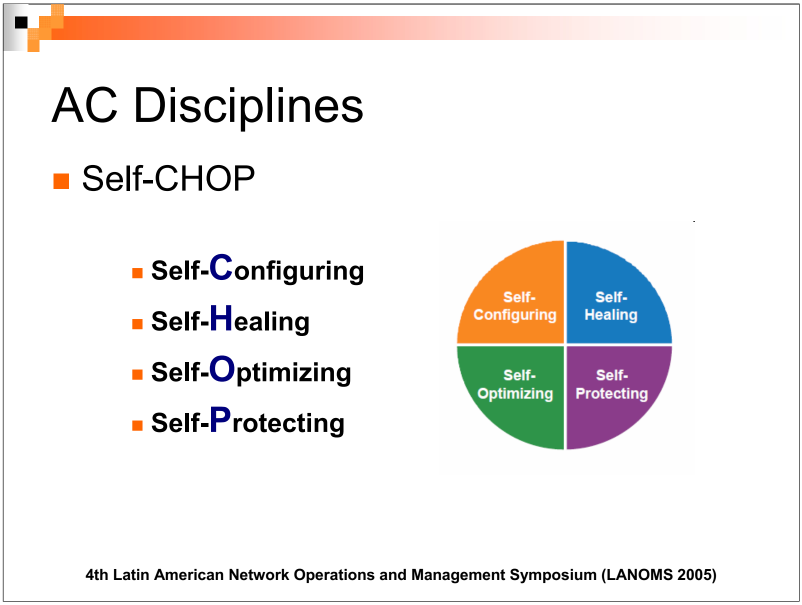
Basic: The product and environment expertise resides in human minds, requiring consultation on even mundane procedures.

Managed: Scripting and logging tools automate routine execution and reporting. Individual specialists review information gathered by the tools to make plans and decisions.

Predictive: Early warning flags are raised as preset thresholds are tripped. The knowledge base recommends appropriate actions. The proposed resolution of events is leveraged by a centralized storage of common occurrences and experience.

Adaptive: Building on the predictive capabilities, the adaptive system takes action itself based on the situation.

Autonomic: Policy drives system activities such as allocation of resources within a prioritization framework.

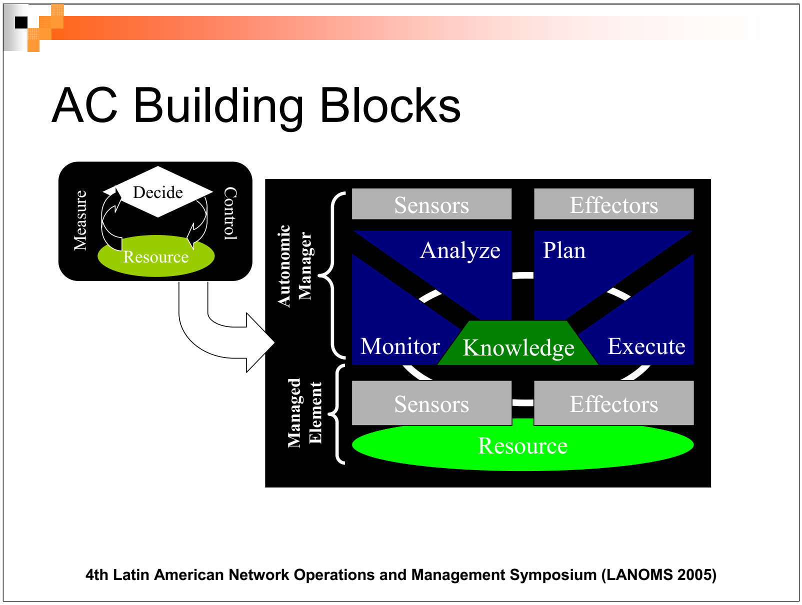


Self-Configuring: Ability to **dynamically configure itself "on the fly"** and initialize itself in the context of the overall system; includes the ability to influence relevant changes in other products in the environment.

Self-Healing: Ability to **recover from a failing component** by first detecting improper operations (either proactively through predictions or otherwise) and then initiating corrective action without disrupting applications.

Self-Optimizing: Ability of systems or components to **efficiently maximize resource allocation and utilization** to meet end-user needs without human intervention.

Self-Protecting: Ability of a component to **detect hostile or intrusive behavior** as it occurs and to take autonomous actions to make itself less vulnerable.



Monitor: Mechanisms to collect, aggregate, filter element details (metrics, topologies, etc)

Analyze: Mechanisms to model and/or analyze complex situations

Plan: Mechanisms to structure the action needed to achieve goals and objectives

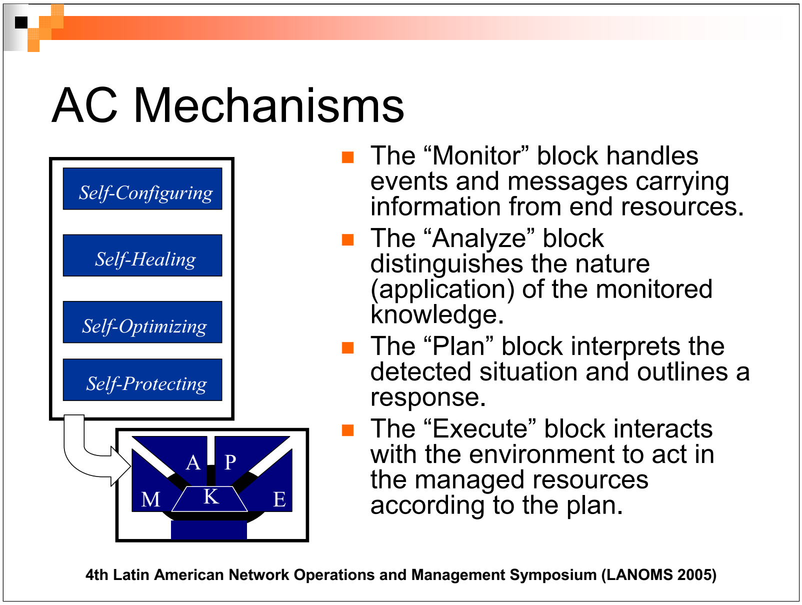
Execute: Mechanisms to execute the actions (plan) with on-the-fly updates

Resource: The entity or entities being managed

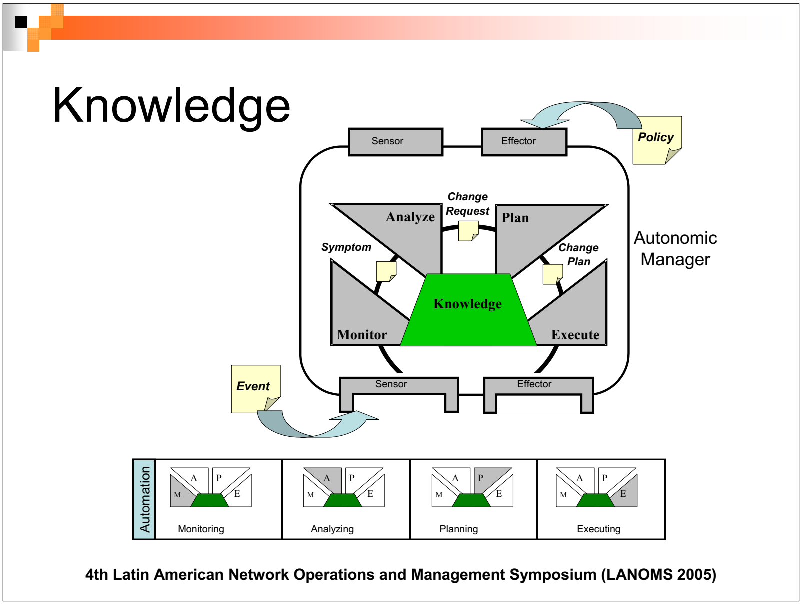
Effectors: Mechanisms to change the state of an element

Sensors: Mechanisms to collect state and state transition information about element

Knowledge: Data and information used and shared among the elements of the MAPE loop



- The **monitor** function provides the mechanisms that collect, aggregate, filter and report details (such as metrics and topologies) collected from a managed resource.
- The **analyze** function provides the mechanisms that correlate and model complex situations (for example, time-series forecasting and queuing models). These mechanisms allow the autonomic manager to learn about the IT environment and help predict future situations.
- The **plan** function provides the mechanisms that construct the actions needed to achieve goals and objectives. The planning mechanism uses policy information to guide its work.
- The **execute** function provides the mechanisms that control the execution of a plan with considerations for dynamic updates.



Several different knowledge formats coexist inside an autonomic manager.

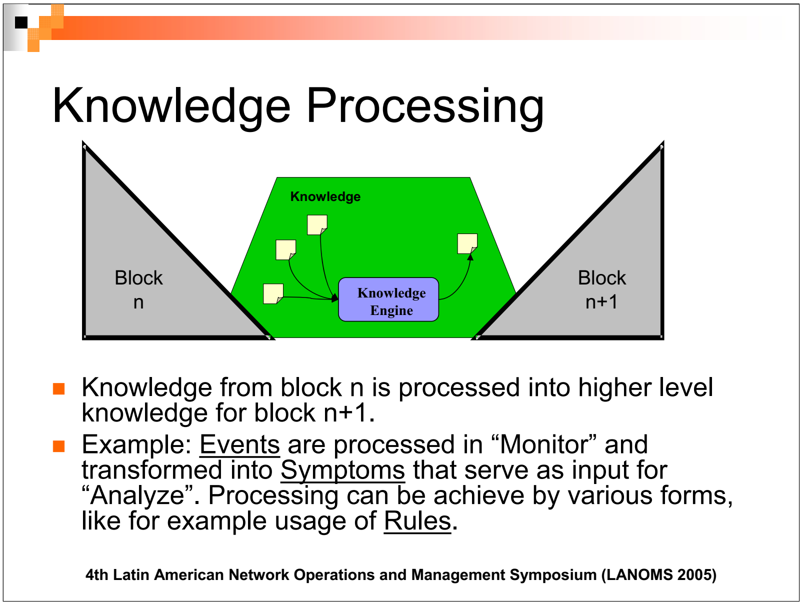
Events are collected from Touchpoints and are processed to form Symptoms.

Symptoms can be combined together until a root cause is determined and Change Requests are produced in response.

Change Requests are combined and processed together to produce Change Plans.

Change Plans are processed and actions are executed in the managed resources.

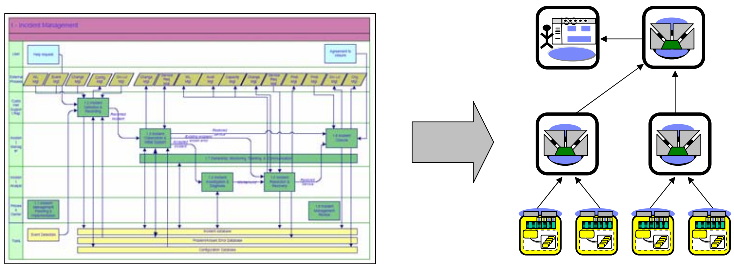
Policies can exist in multiple roles and can be applied together along with all other knowledge forms.



Autonomic Managers operate by continuously processing knowledge. How the knowledge is processed depends upon what functions are desired to be performed by that autonomic manager. Each of the main functions will have **processing artifacts** associated to the types of knowledge they require for performing a determined goal. An example of a processing artifact is a **rule** or a **state machine**.

For example, the monitor layer would receive several different kinds of **events** from different resources. In order to determine if a network failure is responsible by a database failure, we could have a rule that **correlates** the first event (network node down) with the second event (database timeout) and produces a **symptom** (network connectivity down in database node). In upper layers of our autonomic manager, these symptoms can also be correlated together with **policies**, for example, in order to produce **change requests** for the healing of the situation. Eventually these change plans would be turned into **change plans** and combined with environmental knowledge so the healing **actions** can be realized (rebooting a router in the network). This scenario describes a self-healing function.

Autonomic Business Processes



- Business processes have distinct tasks that can be delegated to Autonomic Managers

4th Latin American Network Operations and Management Symposium (LANOMS 2005)

The IT industry focus is to define and understand the various IT processes. The IT landscape is composed by a myriad of services, processes, tasks, procedures, roles, etc, and it amounts to a huge body of knowledge that is not totally understood. New initiatives (like **ITIL©**) aim to define these elements in a library of common processes that can be applied throughout the IT industry. Example: Incident Management, Problem Management, Software Distribution, etc.

Certain aspects of these processes are identified as automation points and are the points where the autonomic computing initiative is applied towards making the system self-aware and ultimately to turning the system into a self-managing entity.

Conclusion

- The Autonomic Computing initiative aims to establish a roadmap so software systems can evolve to be true self-managing systems.
- It is based on “de-facto” and “official” open standards dedicated to foster interoperability and multi-vendor support.
- It aims to be pervasive within the industry.
- It brings together researchers, faculty, industry and businesses towards a common goal.
- It provides incentives for the practical advancement of computer science for years to come. Various problems need to be solved to enable the technology.

4th Latin American Network Operations and Management Symposium (LANOMS 2005)

The Autonomic Computing initiative provides a practical roadmap to achieve true self-managing systems.