Autonomic Computing
Research Landscape

Manish Parashar
The Applied Software Systems Laboratory
Rutgers, The State University of New Jersey
http://automate.rutgers.edu

Salim Hariri
High Performance Distributed Computing Laboratory
The University of Arizona
http://www.ece.arizona.edu/~hpdc

ICAC 2004 Autonomic Computing Tutorial
May 16, 2004

Current Design and Development of Computing Systems

Focused systems evolved separately and targeted specific domains/applications

Secure Military Systems
Cost-Effective Consumer Systems
Fault-Tolerant Financial Systems
High-Performance Scientific Computing Systems
A Convergence of Requirements

High Cost
Unmanageable
Insecure
Interoperability

Autonomic Computing: A Holistic Approach

Cost-Effective Consumer Systems
Secure Military Systems
Fault-Tolerant Financial Systems
High-Performance Scientific Computing Systems

Autonomic Building Block
Self-Healing Component
Self-Optimizing Component
Self-Configuring Component
Self-Protecting Component

Autonomic Computing Systems
Existing Systems

• **Academic Efforts**
  - *University Research*
  - *Work contributing to the Autonomic computing systems beyond IBM’s laboratories.*
  - *Research Projects include*
    - OceanStore - UC Berkeley Computer Science Division
    - Kinesthetics eXtreme (KX) - Columbia University
    - The Horus Project - Cornell University
    - Anthill - Department of Computer Science University of Bologna, Italy
    - Software Rejuvenation – Duke University
    - eBiquity- University of Maryland Baltimore County
    - Recovery Oriented Computing - UC Berkeley / Stanford
    - Autonomia – University of Arizona

Existing Systems

• **Industry Efforts**
  - *IBM committed focus to working within its own global labs and researchers.*
    - Gryphon: Pub/Sub
    - Smart-Self Managing and Resource Tuning DB2
    - Sabio
    - Storage Tank
    - Océano
    - Smart Grid
  - *Microsoft Research*
    - AutoAdmin
OceanStore

- Definition
  - A utility infrastructure designed to span the globe and provide continuous access to persistent information.

System Architecture

- Naming and Access Control
  - GUID (Naming)
  - Reader and Writer Restriction.

- Data Location and Routing
  - Fast Probabilistic Routing Algorithm (Self-Optimizing)
  - Slower, reliable hierarchical routing method (Plaxton Scheme)
    - Self healing, Self optimizing and Self Managing

- Update Model
  - Managed by a series of replica’s
    - Master, Primary and Secondary tier of replicas.

- Durable Storage
  - Active Data in floating replicas.
  - Archival Data in Erasure Coded fragments.
OceanStore Autonomic Features

- **Autonomic and Dynamic Optimization** (Self-optimization)
- **Monitoring and adaptation of routing substrate**
  - Optimization of Plaxton Mesh (Self-configuration)
  - Adaptation of second-tier multicast tree
- **Continuous monitoring of access patterns** (Self-healing)
  - Enhance performance through pro-active movement of data
- **Continuous testing and repair of information** (Self-protection)
  - Automatic replication for disaster recovery
  - Diagnosis and repair of routing and location infrastructure

Columbia’s Programming Systems Lab (PSL)

- How can we construct self-managing, self-configuring, self-healing, self-protecting, context-sensing and continuously self-optimizing systems from legacy components?
- Augment system-of-systems with a decentralized decision & control mechanism that can specify and manage both local optimizations and full-system reconfigurations
  - Decentralized process/workflow definition and enactment
- **Retrofitting Autonomicity**
  - Approach to autonomizing legacy systems and assembling autonomic systems-of-systems
  - Enable autonomic properties through a solution orthogonal to the legacy systems' main business logic and communication framework
- **Common External Infrastructure**
  - Four tiered infrastructure
    - Probes
    - Gauges
    - Coordinated Effectors
    - Architectural Model-based Analysis & Decision
Architecture of the Common Infrastructure

Kinesthetics eXtreme (KX)

- Columbia PSL’s implementation of Common Infrastructure
- KX is being applied for load balancing, etc. in Telecom Italia Lab’s heterogeneous instant messaging and ISI’s open information geographical analysis system (GeoWorlds).
## The Anthill Project

- The Anthill project builds on the similarities between P2P systems and social colonies of ants.
- Anthill construct P2P services that exhibit resilience, adaptation and self-organization properties.
- Ant Colony Algorithms
  - **Agent Based**
    - Artificial Ants of limited individual capabilities move across network of nodes trying to solve a particular problem.
    - While moving they build partial solutions and modify the problem representation by adding collected information.
  - **Complex Adaptive**
    - Individual ants are unintelligent and have no problem solving capability.
    - Nevertheless ant colonies manage to perform several complicated tasks.

## A Parallel between Ant Colonies and Anthill

<table>
<thead>
<tr>
<th>Ant Colonies</th>
<th>Anthill Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real ants need food to survive</td>
<td>Anthill ants need resources to survive (i.e. documents, cpu etc.)</td>
</tr>
<tr>
<td>Environment composed of nests, food sources and land.</td>
<td>Anthill networks composed of nests (which are both nest and food sources) and links between them.</td>
</tr>
<tr>
<td>Real ants move in the environment in the search of food.</td>
<td>Anthill ants move across the network looking for resources.</td>
</tr>
<tr>
<td>Pheromone trails linking nests and food sources.</td>
<td>“Information” trial linking nests and pointing to resources.</td>
</tr>
<tr>
<td>Real ants move food from source to their originating nest</td>
<td>Anthill ants copy a resource from nest to nest</td>
</tr>
</tbody>
</table>

Nest Architecture composed of 3 logical modules
Ant Scheduler, Communication Layer and Resource Manager
Anthill’s Autonomic Properties

- Ants are autonomous agents capable of
  - Moving across a nest network.
  - Interacting with the nest they visit to pursue their goals.
  - Characterized by their algorithm ("species").
  - Behavior of an ant may be
    - non-deterministic (probabilistic).
    - Depends on its algorithm and its current state.

Evolutionary Framework (Autonomic Nature)
- Anthill exploits “nature” metaphor using evolutionary techniques
  - Genetic Algorithms in tuning ant algorithms.
  - Minimization of the total path length traversed by ants.
- Investigates can genetic algorithms be applied at runtime ??
  - Nests could “steal” algorithms and parameters of visiting ants.
  - Crossover and mutation techniques for generating new ants.

Software Rejuvenation

- Proactive fault management technique aimed at preventing crash failures and performance degradation
  - Involves occasionally stopping the running software, “cleaning” it’s internal state and restarting it
  - Counteracts the aging phenomenon
    - Frees up OS resources
    - Removes error accumulation
      - Garbage collection, defragmentation, flushing kernel and file server tables etc.
Granularity of Rejuvenation

**Level 1 rejuvenation**
- Restart service
- Only when stoppage of service saves necessary states

**Level 2 rejuvenation**
- OS reboot
- Application failover and recovery by cluster management software

Approaches to Rejuvenation and Examples

- **Open loop** – periodic
  - No feedback from the system
- **Closed loop** (feedback control)
  - Feedback from the system (monitoring)
- **Applications**
  - AT&T billing applications
  - Software capacity restoration
  - On-board preventive maintenance for long-life deep space missions (NASA's X2000 Advanced Flight Systems Program)
  - Patriot missile system software - switch off and on every 8 hours
  - IBM Director Software Rejuvenation
  - Process Recycling in IIS 5.0

Prediction based Rejuvenation in IBM Director
Recovery-Oriented Computing (ROC)

• Philosophy: “If a problem has no solution, it may not be a problem, but a fact, not to be solved, but to be coped with over time” — Shimon Peres (“Peres’s Law”)
  – People/HW/SW failures are facts, not problems.
  – Recovery/repair is how we cope with them.
• Improving recovery/repair improves availability.
  – UnAvailability = MTTR/MTTF

• Five “ROC Solid” Principles
  – Given errors occur, design to recover rapidly.
  – Given humans make errors, build tools to help operator find and repair problems.
    • e.g., undo; hot swap; graceful, gradual SW upgrade.
  – Extensive sanity checks during operation.
    • To discover failures quickly (and to help debug)
  – Any error message in HW or SW can be routinely invoked, scripted for regression test.
    • To test emergency routines during development.
  – Recovery benchmarks to measure progress.
    • Recreate performance benchmark competition.
• Three R’s for recovery
  – **Rewind:** roll all system state backwards in time.
  – **Repair:** change system to prevent failure.
    • e.g., fix latent error, retry unsuccessful operation, install preventative patch.
  – **Replay:** roll system state forward, replaying end-user interactions lost during rewind.

FIG: Fault Injection in *glibc*

• Objective:
  – Develop a tool for injecting faults at the system boundary
• Motivation:
  – Developers are lazy
  – We need testing tools that generate a wide variety of unexpected faults
• Implementation
  – Thin stub library between app & libc
  – Traps API calls
    • Logs them
    • Inserts faults
  – Can be inserted into any app without modification
    • Uses LD_PRELOAD environment variable.
Pinpoint

- **Motivation**
  - Systems are large and getting larger
  - Systems are dynamic
  - Difficult to diagnose failures

- **Pinpoint Approach Version 1**
  - Trace many real client requests
    - Record every component used in request
    - Detect success/failure of requests
    - Can be used as dynamic dependency graphs
  - Statistical Correlation
    - Search for components that “cause” failures
    - Built into middleware
      - Requires no application code changes
      - Application knowledge only for end-to-end failure detection

- **Implementation**
  - Built on top of J2EE platform
  - Version 2 of Pinpoint
    - instruments JBoss middleware.
    - Modify HTTP server, wrappers for EJB, JSP, JDBC
    - Observe calls to and returns from components, exceptions
    - Record component details, SQL queries, timestamp
    - Record path context: request id, sequence number

UMBC eBiquity Research Group

- Explores the interactions between mobile, pervasive computing, multi-agent systems and artificial intelligence, and e-services.
- Few Projects
  - Agents2Go
  - Anamika: Service Composition in Ad-hoc Environments
  - OWLIR: Information Retrieval On The Semantic Web
  - Allia
  - An Agent Based Distributed Computing System using XReggie and Ronin
  - SweetJess
  - UMBC OntoMapper: A Tool For Mapping Between Two Ontologies
  - Learning to Tag: Generating DAML mark up for Semistructured Documents
  - MoGATU: Profile-Driven Data Management in Pervasive Environments
Agents2Go

- Infrastructure Location-Dependent Service Discovery in Mobile Electronic Commerce.
- Automatically obtains a user’s current geographical location in CDPD (Cellular Digital Packet Data) without relying on GPS.
- Communities of software agents called agencies provide information services and e-commerce support.
- Components of the Agents2Go:
  - PalmApp
  - Agents2Go Server:
    - Centaurus Communication Protocol (CComm)
  - Locator
  - Agents2Go Information Repository
  - Restaurant Brokers
  - Participating Restaurant Agents.

Anamika

- Reactive Service Composition for Pervasive Computing Environments
- Components
  - Network Layer
    - WLAN (802.11) networks
  - Service Discovery Layer
    - Peer to Peer service discovery
      - Semantic description based service matching (DAML + OIL)
  - Dynamic Caching of neighboring service description
  - Service Composition Layer
    - DAMLS
  - Service Execution Layer
  - Application Layer
- Dynamic Broker Selection Technique
  - Broker Arbitration and Delegation
  - Service Integration and Execution
  - Fault Recovery

User Interaction with the Anamika System

Ad-hoc service Composition environment

System Components in the Anamika
Reactive Service Composition
**OWLIR**

- Approach to retrieval of documents containing free text and semantically enriched markup.
- OWLIR consists:
  - Set of ontologies
    - Ontology explicit specification of a representational vocabulary for shared domain of discourse.
  - Event Ontology
    - Extension of ITTalks ("Natural Kinds of")
- Text Extraction
  - AeroText System
  - Extracts key phrases & elements from free text docs.
- Inference System
  - DAMLJessKB reads DAML + OIL files.
- Information retrieval
  - Hopkins Automated Information Retriever for Combining Unstructured Text (HAIRCUT).
  - Document modeling approach to reasoning document similarity.
  - Word or N-gram based Dynamic Information Retrieval Engine (WONDIR).

```xml
<Query (DAML query)
<required>
triple(query_001)(
'http://gentoo.cs.umbc.edu/howlir/query#query_001',
'http://gentoo.cs.umbc.edu/howlir/ontologies/event_ont#Movie_Name'
'Ocean Eleven').
</required>
<allowed></allowed>
<disallowed>
'http://gentoo.cs.umbc.edu/howlir/query#query_001',
'http://gentoo.cs.umbc.edu/howlir/ontologies/event_ont#Organizer'
'SEB').
</disallowed>
</Query>
```

**Sabio**

- Sabio takes large collections of documents within an enterprise and breaks them down automatically into a taxonomy.
  - unaided by human categorizers.
  - Automated Taxonomy Generator as it is called in Raven.
- employs Bayesian statistics
  - educated guess
- decompose each document into a collection of "tokens".
- assembles a collection of relevant words and phrases in all the documents.
  - treats this collection mathematically as points in a huge multidimensional space.
    - each dimension corresponds to a single word or phrase,
  - number of times the word or phrase appears determines how far out along the dimension the point lies.
- 2 documents which share many of the same words and phrases relatively close together in this multidimensional space.
- Combined with the Lotus Product
  - ability to bring order to the most chaotic set of corporate e-documents.
Horus Project

- A framework for development of distributed applications based on group communications.
  - redesign of the Isis group communication system
  - fault-tolerant systems
  - managed distributed systems
  - applications that exploit data replication or coherent caching, and groupware.
- Virtual synchrony
  - Runtime model for data replication and fault-tolerance
- Ensemble
  - New version of Horus written in ML.
  - will be an outstanding environment for building Java-based groupware applications that do multimedia conferencing on the Web.

Architecture

Group Protocol Layers can be stacked at runtime as Lego Blocks
Support variety of applications through “interface proxy”

Object Group Communication in Electra
Electra is a CORBA’s Object Request Broker (ORB)
Plug and Play Approach to Group Computing

Unix system calls are intercepted by Horus using an intercept proxy
**Gryphon**

- Publish/subscribe middleware aimed at distributing large volumes of data in real time.
- **Features**
  - topic based and content based publish/subscribe
  - publish/subscribe system deployed on a public network cannot depend on homogenous router technology.
    - Use of tcp/ip or http.
  - Scale support to application growth.
  - Provide security and privacy features to a degree not mandated over private secured networks.
  - Client authentication, access controls and encryption/integrity of messages
- **Implementation**
  - Java Message Services (JMS) API.
  - Patented Matching Engine provides high speed content filtering.

**SMART DB2**

- The DB2 SMART project aims to create technology for reducing human intervention and cost in DB2 operation.
- It builds on and extends existing self-managing technologies in DB2.
  - Adjust every configuration parameter dynamically while the system is in use
  - Expand and shrink memory usage, based on workload
  - Automatically profile workloads and recommend and create indexes, partitioning, clustering, summary tables, and so on to improve performance
  - Automatically detect the need for, estimate the duration of, and schedule maintenance operations such as reorganization, statistics collection, backup, copy, and rebinding
  - Observe actual performance and exploit that information to improve operations
  - Recommend action when the performance isn’t meeting the DBA’s expectations
  - Predict problems such as low memory or constrained disk space and notify someone by pager or email in advance.
DB2’s Autonomic Features

- **Query Optimizer**
  - Automatically determines the best way to execute a declarative SQL query.
- **Automatic selection of degree of parallelism**
  - Setting and adjusting degree of parallelism for queries and utilities.
- **Detection of partial disk writes.**
  - Protects data integrity by automatically detecting any corrupted data from incomplete I/O’s.
- **Application Control and Tuning**
  - **Query Patroller**
    - “Predictive Governer” uses the “Query Optimizer” estimate of relative resources for each query to limit surges of arriving or long running queries.
    - “Reactive Governer” monitors the actual resources consumed to prevent runaway queries.
  - **Performance Expert.**
    - Performs passive monitoring and collects trace and monitor data in a performance data warehouse.
    - **Buffer Pool Analyzer**
      - Collects buffer pool activity and models changes to the objects in the buffer pools.

AutoAdmin

- **Self Tuning and Self Administering Databases.**
- **Enabling databases to track the usage of their systems and to gracefully adapt to application requirements**
- **Bottom up approach**
  - Choose appropriate physical objects and their organization
    - **Indexes**
    - **Materialized Views**
    - **Statistics**
  - Goal: Optimize performance
Astrolabe

- Astrolabe is an information management service
  - Virtual system wide hierarchical database evolves as the underlying information changes.
    - Relational database built using peer-to-peer protocol.
    - Ability to perform data mining and data fusion
    - Continuously computes summaries using on-the-fly aggregation.
  - Self configuration, distributed monitoring and control adaptation.

Architecture

<table>
<thead>
<tr>
<th>Name</th>
<th>Load</th>
<th>WebLogic</th>
<th>SMTP</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>astol</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>astol</td>
<td>0.5</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>astol</td>
<td>4.5</td>
<td>1.0</td>
<td>0.0</td>
<td>5.0</td>
</tr>
</tbody>
</table>

Collection of small database relations, each tuple corresponding to one machine.

Each relation collecting tuples associated with some set of nearby machines.

“Aggregation Query” SQL query operates on leaf operations, extracting a single summary tuple from each which reflects a globally significant information within the region.

Storage Tank

- SAN based multiplatform distributed file system and storage management solution
  - Policy based storage and data management.
  - Shared heterogeneous (multiplatform) data sharing.
  - Exploitation and enablement of SAN technology.
  - Massive scalability of data, servers and clients.
  - High Availability of servers and data.
  - Global namespace and single system semantics.

- Storage Tank abstractions
  - Container
    - Sub tree of the global namespace
  - Storage Pool
    - Collection of one or more disk volumes.

- Storage Tank Protocol
  - Aggressive caching at client
  - Retention of modified meta-data, data and lock state beyond close-of-file by an application.

- Storage Tank Client
  - File System Interface
  - Client State Manager
  - Operating System Services.
Océano Project

- "Computing Utility Powerplant"
- "FARM" of massively parallel, densely-packaged servers interconnected by high-speed, switched LANs.
- High levels of automation to dynamically adjust web sites to actual traffic demands.
- Implements infrastructure enabling large numbers of hosted customers
- Reduce the costs of setting up and operating the hosting farms by automation
- Objectives
  - Dynamically assign resources to accommodate planned and unplanned fluctuation of workloads
  - Offer a wide variety of services levels to customers
  - Secure sharing of resources across multiple customers
  - Provide adequate reliability through massive redundancy, and automated re-provisioning.
- Resource allocation
  - Collects and utilizes SLA data
  - Monitors and processes SLA policies
  - Provides a scalable framework for monitoring resource status
  - Provides application metric collectors for some standard e-commerce software packages.

Autonomia

- AUTONOMIA environment provides the application developers
  - Tools required to specify the appropriate control and management schemes
    - to maintain any quality of service requirement
    - application attribute/functionality (e.g., performance, fault, security, etc.)
  - Core autonomic middleware services
- Self-Configuring Engine
  - Responsible for configuring/reconfiguring the applications on the air.
  - Chooses the appropriate policy specified by the self-configuring profile to configure the application.
- Self Optimizing
  - Optimizes application as well as system performance at runtime
  - Handler selects appropriate mechanism to optimize application performance
- Self-Protecting Handler
  - Uses the idea of intention list to make decisions on the fly about access control to various tasks.
  - Autonomia Security Manager constantly monitors the agent intention list and tasks.
  - Agent only allowed to execute tasks published in the intention list.
  - Any deviation causes loss of further access for the agent.
Smart Grid

- OptimalGrid is a project in the distributed systems department at the IBM Almaden Research Center designed to solve the next generation of large scale parallel problems on a large number of network-attached, heterogeneous compute nodes (i.e. "The Grid").
- OptimalGrid automates aspects of solving a large scale "connected problem" on a computing Grid,
- To enable this Grid compute utility model
  - **Autonomic Load Balancing.**
    - Adapt problem units to dynamic changes in available computing resources
    - Manage correlations between the problem units
  - **Establish micro-billing mechanisms.**
Attacking an FEM Grid Problem

- Domain Expert partitions problem (partitions space) into “Original Problem Cells” (OPCs)
  - Data
  - Methods
  - Neighbor pointers for inter-cell interactions
    - Position
    - State
- Collection of OPCs is grouped into a “Compute Unit” called a Variable Problem Partition (VPP)

Smart Grid Prototype

- Major components:
  - Autonomic Program Manager (APM)
  - Variable Problem Partitions (VPP) (collections of OPCs)
  - Computing Agents (CA)
  - Autonomic Rule Engine (ARE)
  - Micropayment Broker (MPB)
  - UDDI Server (Universal Description Discovery Integration)
  - OSGi (Open Services Gateway Initiative)
- Component Roles:
  - APM employs the ARE and manages CAs
  - CAs run VPPs, communicating with other VPPs (CAs) thru some mechanism.
  - CAs log performance data that is used by the ARE to adjust the VPP sizes (allocations) for each of the CAs
## Summary of Autonomic Computing Systems

### Autonomic Computing Systems

<table>
<thead>
<tr>
<th>Autonomic Computing Properties</th>
<th>OceanStore</th>
<th>KX</th>
<th>Astrolabe</th>
<th>AntHill</th>
<th>ROC</th>
<th>eBiQuity</th>
<th>Software Rejuvenation</th>
<th>Autonomia</th>
<th>Automate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self Optimizing</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Self Healing</td>
<td>✔</td>
<td>☒</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>☒</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Self Protection</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>☒</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Context Aware</td>
<td>☒</td>
<td>☒</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>☒</td>
<td>☒</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Self Configuring</td>
<td>☒</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>☒</td>
<td>☒</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Self Regulating</td>
<td>☒</td>
<td>☒</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>☒</td>
<td>☒</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Open</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>☒</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Self Awareness</td>
<td>☒</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

= supported, ✗ = not supported

### Research Institutions (Industry)

<table>
<thead>
<tr>
<th>Autonomic Computing Properties</th>
<th>Sabio</th>
<th>Smart</th>
<th>Océano</th>
<th>Optimal Grid</th>
<th>Gryphon</th>
<th>Storage Tank</th>
<th>AutoAdmin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self Optimizing</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Self Healing</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Self Protection</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Context Aware</td>
<td>✗</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Self Configuring</td>
<td>✗</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Self Regulating</td>
<td>✗</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Open</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Self Awareness</td>
<td>✗</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

= supported, ✗ = not supported
Conclusions

- Existing Systems can be divided into
  - Systems which Address Autonomic properties
    - SMART
    - AutoAdmin
    - Anthill
    - Software Rejuvenation
  - Help to build systems which address these properties.
    - eBiquity’s Research
    - Gryphon
    - Columbia’s Programming Systems Lab (PSL)
    - The Horus Project
    - Autonomia
- Systems address specific issues of the 8 elements of Autonomic Computing.

References

- “Software Rejuvenation,” http://shannon.ee.duke.edu/Rejuv/
References Contd..