**Existing Systems**

- **Academic Efforts**
  - University Research
  - Work contributing to the Autonomic computing systems beyond IBM’s laboratories.
  - Few 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

- **Industry Efforts**
  - IBM committed focus to working within its own global labs and researchers.
    - Gryphon: Pub/Sub
    - Smart-Self Managing and Resource Tuning DB2
    - Sakti
    - Storage Tank
    - Ocobano
    - 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)
- **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 Autonomity
  - 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

- architectural models
- probes
- gauges
- coordinated effectors
- decision
- probe bus
- gauge bus
- architectural model-based analysis & decision

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>What ants need food to survive</td>
<td>Anthill ants need resources to survive (data, documents, etc.)</td>
</tr>
<tr>
<td>Environment consists of nests, food sources and links</td>
<td>Anthill infrastructure composed of self-organizing overlay network of nests (N1 to N6)</td>
</tr>
<tr>
<td>What ants move in the environment to find food</td>
<td>Anthill ants move across the network looking for resources</td>
</tr>
<tr>
<td>Forage for food, finding nests and food sources</td>
<td>Anthill infrastructure composed of nests (N1 to N6) and links between them</td>
</tr>
<tr>
<td>What ants move from source to their originating nest</td>
<td>Anthill ants copy a resource from nest to its local nest</td>
</tr>
</tbody>
</table>

Nest Architecture composed of 3 logical modules
- Ant Scheduler
- Communication Layer
- 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 ??
  - Neats could “idea” algorithms and parameters of existing 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" its internal state and restarting it
  - Counteracts the aging phenomenon
- Frees up OS resources
- Removes error accumulation
  - Garbage collection, defragmentation, flushing server tables etc.

Software Rejuvenation reduces downtime

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
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/Media 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.
  - Exception using PINPOINT tool during test.
  - In the absence of failing point in the deployed system, any occurrence in NW or SW can be remotely invoked, scripted for regression test.
  - To test emergency routines during development.
- Recovery benchmarks to measure progress.
  - Demonstrate performance benchmark completion.
- Three R’s for recovery
  - Re-knit: not all system state backdrops in time.
  - Repay: change system to prevent failure.
  - Repop: roll system state forward, replaying end-user interactions lost during rewind.

FIG: Implementation

- 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
  - This 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.

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
  - This 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.

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
  - This 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
  - Traces many real client requests
  - Record every component used in request
  - Detect access failures of requests
  - Can be used as dynamic dependency graphs
  - Statistical Correlation
    - Search for components that “cause” failures
    - Build into middleware
    - Requires no application code change
    - Assumes knowledge only for end-to-end failure analysis
- Implementation
  - Built on top of JEE platform
  - Version 2 of Pinpoint
    - Instrumented JBoss middleware
    - Monitors HTTP server, containers for JSP, J2EE
    - Observe code and state within container
    - Detect component details, SQL queries, transactions
    - Detect path taken, request at sequence number

FIG: Fault Injection in glibc
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
  - OWIR: Information Retrieval On The Semantic Web
  - Alia
  - 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)
  - Locato
  - 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
    - DAML-S
  - 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
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 ITTalk ("Natural Kinds of")
  - Text Extraction
    - AeroText System
    - EventOnt Process
  - Information retrieval
    - Hopkins Automated Information Retriever for Combing Unstructured Text (HAIRCUT)
    - Word or N-gram based Dynamic Information Retrieval Engine (WONDIR)

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

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.
  - 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 reload
  - 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 Patrolier**
  - "Predictive Governer" uses the "Query Optimizer" estimate of relative resources for each query to help prevent or 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
  - 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.**
  - "Aggregation Query" SQL query operates on set 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
  - High speed, scalable and reliable
  - Shared homogeneous (multipathing) disk connectivity
  - Flexibility and scalability of SAN technology
  - High availability of servers and data
  - Global namespace and single system image

- Storage Tank abstractions
  - Container
  - Storage Pool

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

- 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

- 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
  - 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 applications as well as system performance at runtime.

- Self-Protecting Handler
  - Uses the ideas 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) through 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**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Software</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AutoAdmin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOD</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>

- **Properties**
  - Self Awareness
  - Self Regulating
  - Self Configuring
  - Context Aware
  - Self Protection
  - Self Healing
  - Self Optimizing

- **Academic Institutions**
  - University of California
  - Stanford University
  - University of Texas
  - University of Illinois

- **Research Institutions (Industry)**
  - Open Source
  - Micropayment
  - Autonomic Rule Engine
  - Micropayment Broker
  - UDDI Server
  - OSGi

---

**Summary of Autonomic Computing Systems**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoAdmin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOD</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>

- **Properties**
  - Self Awareness
  - Self Regulating
  - Self Configuring
  - Context Aware
  - Self Protection
  - Self Healing
  - Self Optimizing

- **Academic Institutions**
  - University of California
  - Stanford University
  - University of Texas
  - University of Illinois

- **Research Institutions (Industry)**
  - Open Source
  - Micropayment
  - Autonomic Rule Engine
  - Micropayment Broker
  - UDDI Server
  - OSGi
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 could address specific issues of the 8 Elements of Autonomic Computing.

References