Towards Autonomic Computing

Ph.D. Thesis Defense

Alexander V. Konstantinou

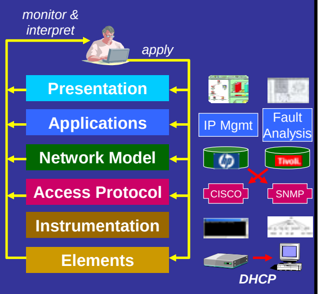
DCC Laboratory
Computer Science Department
Columbia University

Autonomic Computing

- Autonomic computing
  - Important challenge of this decade (management over 70% TCO)
  - Goal: self-configuring/optimizing/healing/protecting systems
- Thesis contributions
  - Peer-to-peer autonomic management architecture
  - Language for embedding autonomic management functions at design time
  - Change propagation model, language, and analysis
  - Autonomic platform prototype implementation (released)
  - Applications: security, service & user mobility, active networks

Autonomy as an Afterthought

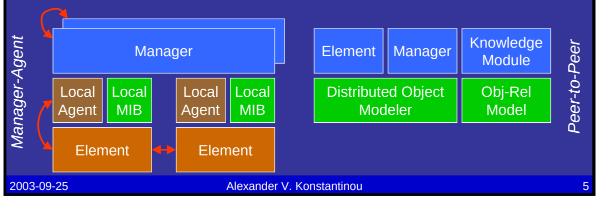
- Traditional management
  - Man-in-the loop
  - Knowledge is diffused
  - Ad-hoc processes & architecture
  - Unsafe & insecure
- Challenges of Autonomy
  - Knowledge distribution
  - Processes & architecture to effect knowledge
  - Safety & security



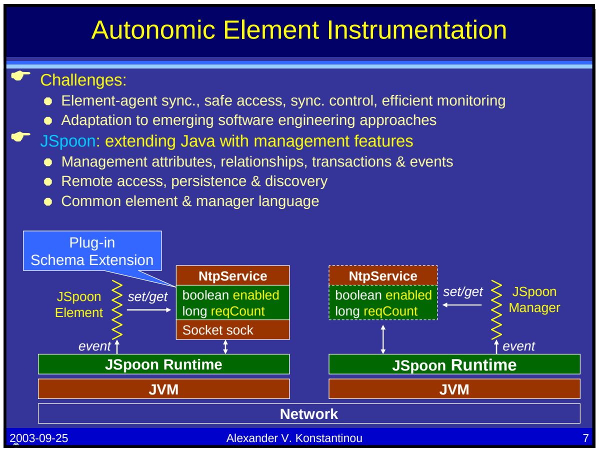
An Architecture for Autonomy

Peer-to-Peer Autonomic Mgmt Architecture

- Two-layered approach:
  - Data modeling layer
    - Unified object-relationship schema
    - Transactional access, event notification, persistence, security
  - Autonomic management layer
    - Semantic schema extensions
- Advantages:
  - Scalability: unified model, multi-manager, publish-subscribe, cross-domain
  - Reliability: safe access, synchronous policy enforcement, reduced complexity



A Language for Autonomy



Autonomic Element Instrumentation

- Challenges:
  - Element-agent sync., safe access, sync. control, efficient monitoring
  - Adaptation to emerging software engineering approaches
- JSpoon: extending Java with management features
  - Management attributes, relationships, transactions & events
  - Remote access, persistence & discovery
  - Common element & manager language

Plug-in Schema Extension

JSpoon Element

set/get

event

NtpService

boolean enabled
long reqCount

Socket sock

JSpoon Runtime

JVM

NtpService

boolean enabled
long reqCount

set/get

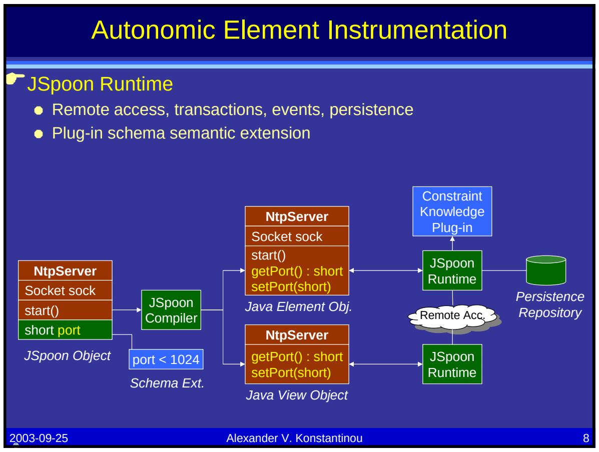
JSpoon Manager

event

JSpoon Runtime

JVM

Network



Autonomic Element Instrumentation

- JSpoon Runtime
  - Remote access, transactions, events, persistence
  - Plug-in schema semantic extension

NtpServer

Socket sock

start()

short port

JSpoon Object

JSpoon Compiler

port < 1024

Schema Ext.

NtpServer

Socket sock

start()
getPort() : short
setPort(short)

Java Element Obj.

NtpServer

getPort() : short
setPort(short)

Java View Object

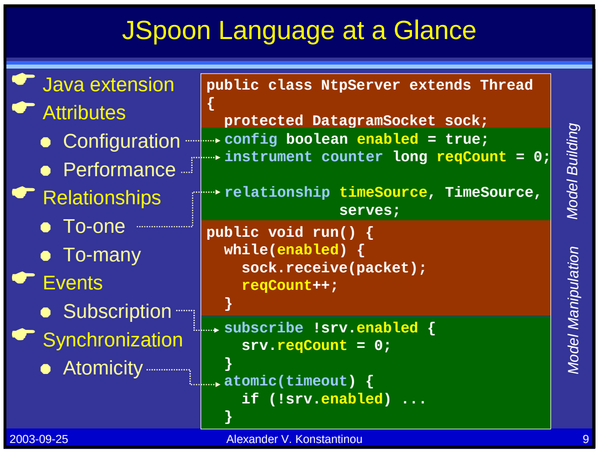
Constraint Knowledge Plug-in

JSpoon Runtime

Remote Acc.

JSpoon Runtime

Persistence Repository



JSpoon Language at a Glance

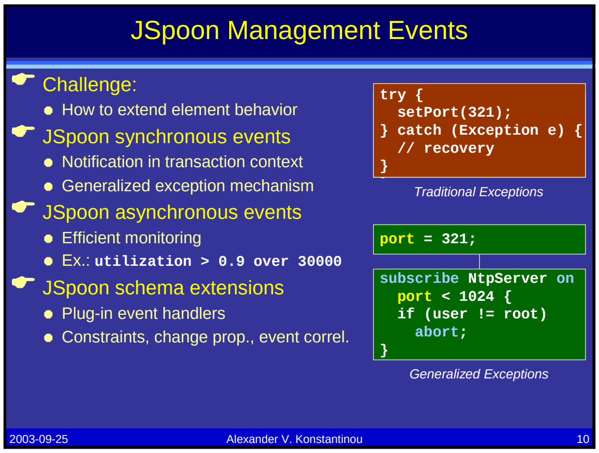
- Java extension
- Attributes
  - Configuration
  - Performance
- Relationships
  - To-one
  - To-many
- Events
  - Subscription
- Synchronization
  - Atomicity

```
public class NtpServer extends Thread
{
  protected DatagramSocket sock;
  config boolean enabled = true;
  instrument counter long reqCount = 0;

  relationship timeSource, TimeSource,
      serves;
public void run() {
  while(enabled) {
    sock.receive(packet);
    reqCount++;
  }
  subscribe !srv.enabled {
    srv.reqCount = 0;
  }
  atomic(timeout) {
    if (!srv.enabled) ...
  }
```

Model Building

Model Manipulation



JSpoon Management Events

- Challenge:
  - How to extend element behavior
- JSpoon synchronous events
  - Notification in transaction context
  - Generalized exception mechanism
- JSpoon asynchronous events
  - Efficient monitoring
  - Ex.: `utilization > 0.9 over 30000`
- JSpoon schema extensions
  - Plug-in event handlers
  - Constraints, change prop., event correl.

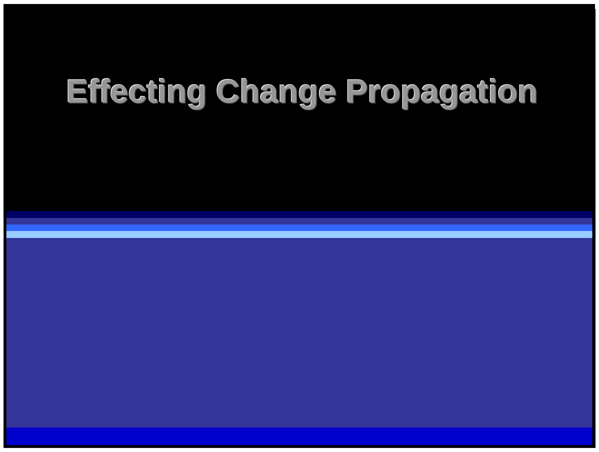
```
try {
  setPort(321);
} catch (Exception e) {
  // recovery
}
```

Traditional Exceptions

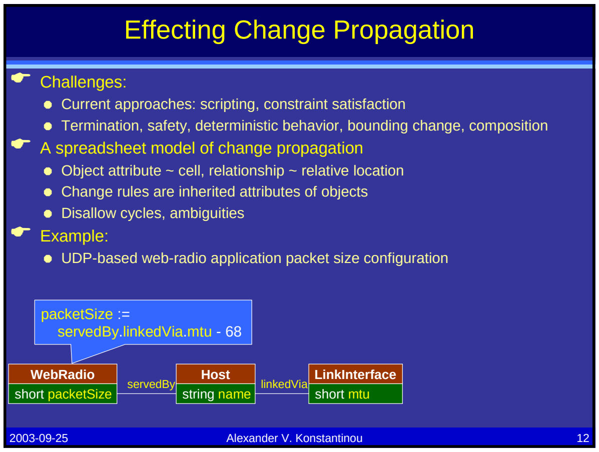
```
port = 321;
```

```
subscribe NtpServer on
  port < 1024 {
  if (user != root)
    abort;
}
```

Generalized Exceptions



Effecting Change Propagation



Effecting Change Propagation

- Challenges:
  - Current approaches: scripting, constraint satisfaction
  - Termination, safety, deterministic behavior, bounding change, composition
- A spreadsheet model of change propagation
  - Object attribute ~ cell, relationship ~ relative location
  - Change rules are inherited attributes of objects
  - Disallow cycles, ambiguities
- Example:
  - UDP-based web-radio application packet size configuration

packetSize :=
servedBy.linkedVia.mtu - 68

WebRadio

short packetSize

servedBy

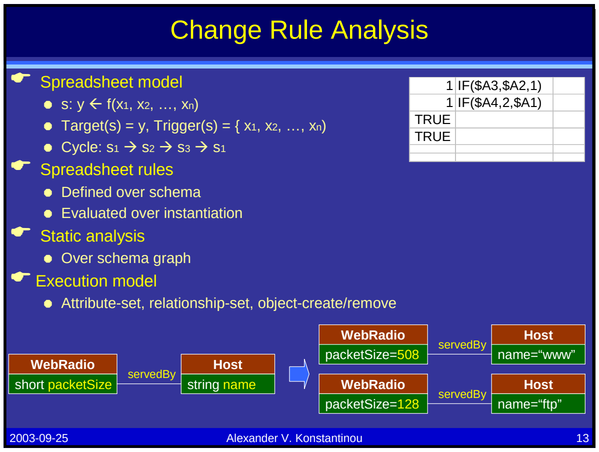
Host

string name

linkedVia

LinkInterface

short mtu

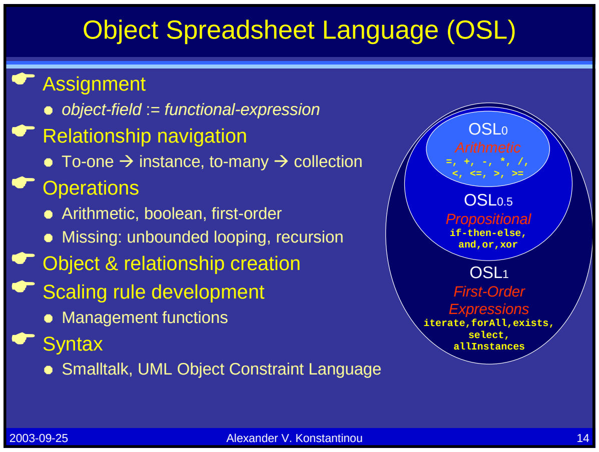


Change Rule Analysis

- Spreadsheet model
  - s: $y \leftarrow f(x_1, x_2, \ldots, x_n)$
  - Target(s) = y, Trigger(s) = { $x_1, x_2, \ldots, x_n$)
  - Cycle: $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_1$
- Spreadsheet rules
  - Defined over schema
  - Evaluated over instantiation
- Static analysis
  - Over schema graph
- Execution model
  - Attribute-set, relationship-set, object-create/remove

1	IF($A3,$A2,1)	
1	IF($A4,2,$A1)	
TRUE		
TRUE		

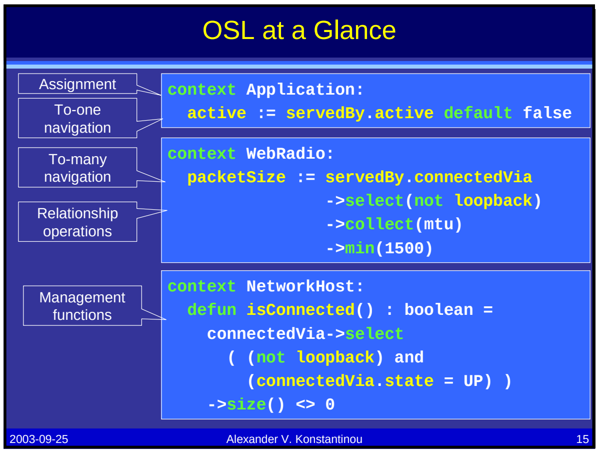
2003-09-25 Alexander V. Konstantinou 13



Object Spreadsheet Language (OSL)

- Assignment
  - *object-field := functional-expression*
- Relationship navigation
  - To-one → instance, to-many → collection
- Operations
  - Arithmetic, boolean, first-order
  - Missing: unbounded looping, recursion
- Object & relationship creation
- Scaling rule development
  - Management functions
- Syntax
  - Smalltalk, UML Object Constraint Language

2003-09-25 Alexander V. Konstantinou 14



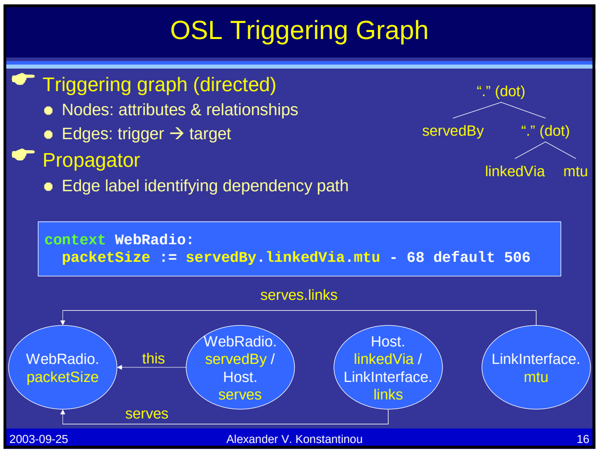
OSL at a Glance

```
context Application:
  active := servedBy.active default false
```

```
context WebRadio:
  packetSize := servedBy.connectedVia
                ->select(not loopback)
                ->collect(mtu)
                ->min(1500)
```

```
context NetworkHost:
  defun isConnected() : boolean =
    connectedVia->select
      ( (not loopback) and
        (connectedVia.state = UP) )
    ->size() <> 0
```

2003-09-25 Alexander V. Konstantinou 15

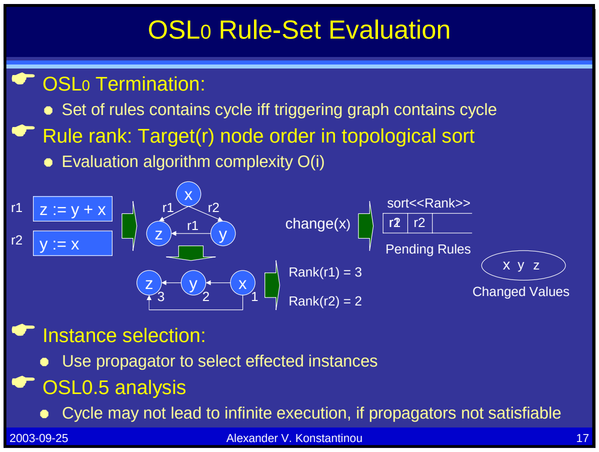


OSL Triggering Graph

- Triggering graph (directed)
  - Nodes: attributes & relationships
  - Edges: trigger → target
- Propagator
  - Edge label identifying dependency path

```
context WebRadio:
  packetSize := servedBy.linkedVia.mtu - 68 default 506
```

2003-09-25 Alexander V. Konstantinou 16

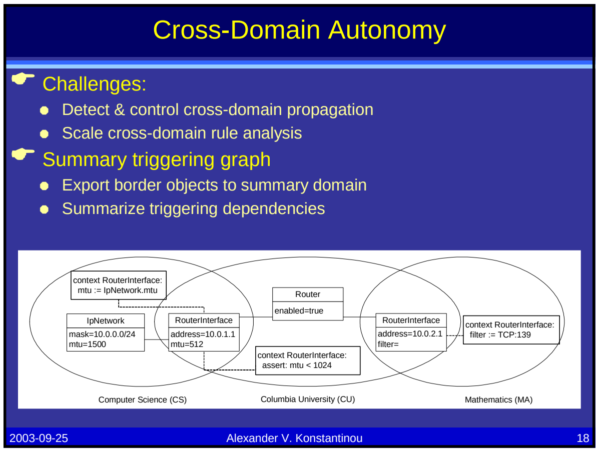


OSL_0 Rule-Set Evaluation

- OSL_0 Termination:
  - Set of rules contains cycle iff triggering graph contains cycle
- Rule rank: Target(r) node order in topological sort
  - Evaluation algorithm complexity O(i)

- Instance selection:
  - Use propagator to select effected instances
- OSL0.5 analysis
  - Cycle may not lead to infinite execution, if propagators not satisfiable

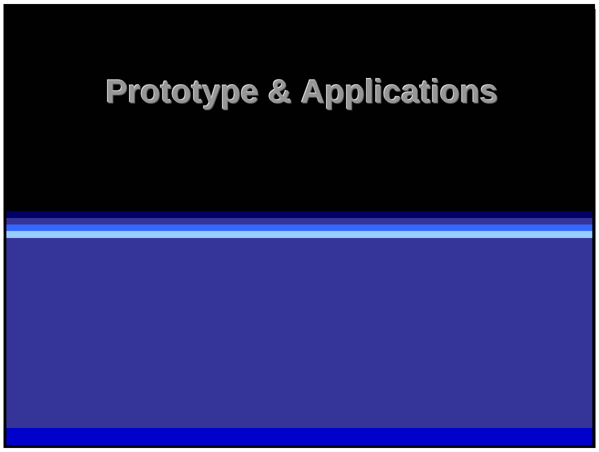
2003-09-25 Alexander V. Konstantinou 17



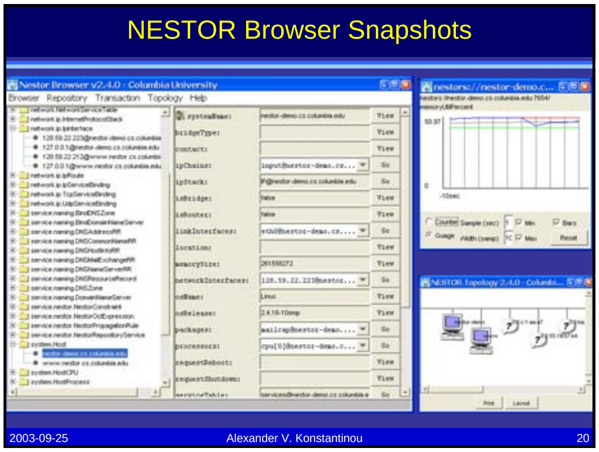
Cross-Domain Autonomy

- Challenges:
  - Detect & control cross-domain propagation
  - Scale cross-domain rule analysis
- Summary triggering graph
  - Export border objects to summary domain
  - Summarize triggering dependencies

2003-09-25 Alexander V. Konstantinou 18



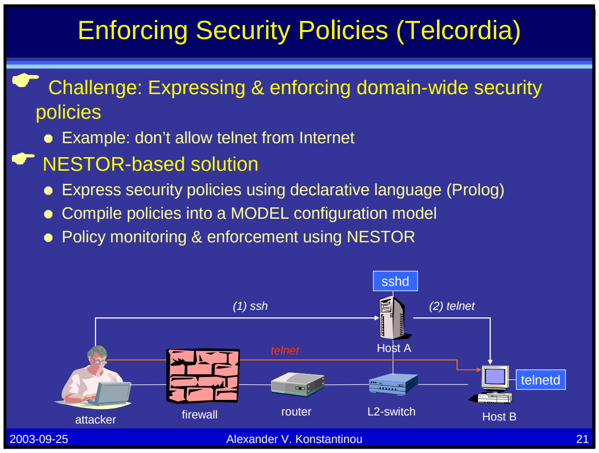
Prototype & Applications



NESTOR Browser Snapshots

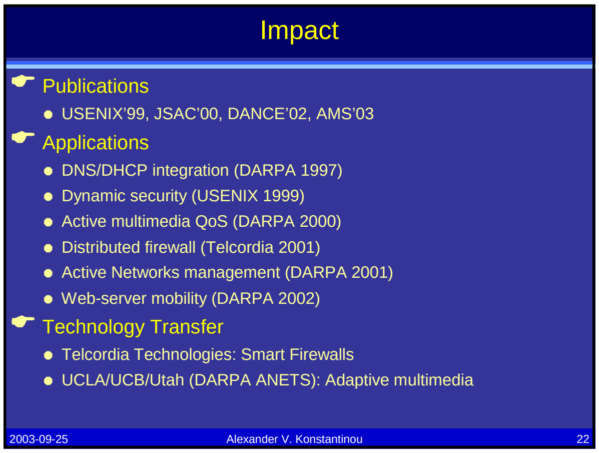
Nestor Browser v2.4.0 - Columbia University
Browser Repository Transaction Topology Help
network.NetworkServiceTable
network.ip.InternetProtocolStack
network.ip.IpInterface
128.59.22.223@nestor-demo.cs.columbia
127.0.0.1@nestor-demo.cs.columbia.edu
128.59.22.212@www.nestor.cs.columbia
127.0.0.1@www.nestor.cs.columbia.edu
network.ip.IpRoute
network.ip.IpServiceBinding
network.ip.TcpServiceBinding
network.ip.UdpServiceBinding
service.naming.BindDNSZone
service.naming.BindDomainNameServer
service.naming.DNSAddressRR
service.naming.DNSCommonNameRR
service.naming.DNSHostInfoRR
service.naming.DNSMailExchangeRR
service.naming.DNSNameServerRR
service.naming.DNSResourceRecord
service.naming.DNSZone
service.naming.DomainNameServer
service.nestor.NestorConstraint
service.nestor.NestorOclExpression
service.nestor.NestorPropagationRule
service.nestor.NestorRepositoryService
system.Host
nestor-demo.cs.columbia.edu
www.nestor.cs.columbia.edu
system.HostCPU
system.HostProcess

systemName: nestor-demo.cs.columbia.edu View
bridgeType: View
contact: View
ipChains: input@nestor-demo.cs.... Go
ipStack: IP@nestor-demo.cs.columbia.edu Go
isBridge: false View
isRouter: false View
linkInterfaces: eth0@nestor-demo.cs.... Go
location: View
memorySize: 261558272 View
networkInterfaces: 128.59.22.223@nestor... Go
osName: Linux View
osRelease: 2.4.18-10smp View
packages: mailcap@nestor-demo.... Go
processors: cpu[0]@nestor-demo.c... Go
requestReboot: View
requestShutdown: View
serviceTable: services@nestor-demo.cs.columbia.e Go



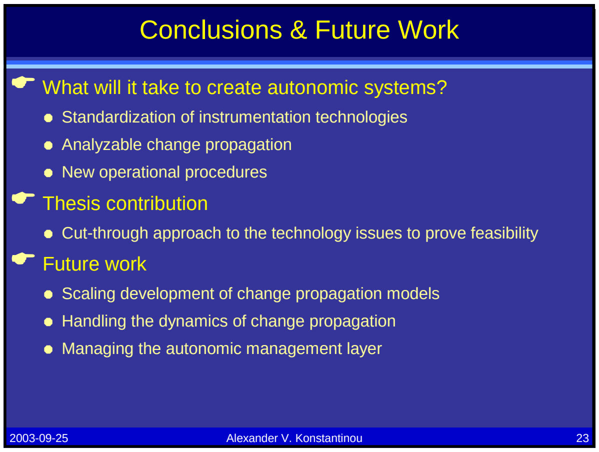
Enforcing Security Policies (Telcordia)

- Challenge: Expressing & enforcing domain-wide security policies
  - Example: don't allow telnet from Internet
- NESTOR-based solution
  - Express security policies using declarative language (Prolog)
  - Compile policies into a MODEL configuration model
  - Policy monitoring & enforcement using NESTOR



Impact

- Publications
  - USENIX'99, JSAC'00, DANCE'02, AMS'03
- Applications
  - DNS/DHCP integration (DARPA 1997)
  - Dynamic security (USENIX 1999)
  - Active multimedia QoS (DARPA 2000)
  - Distributed firewall (Telcordia 2001)
  - Active Networks management (DARPA 2001)
  - Web-server mobility (DARPA 2002)
- Technology Transfer
  - Telcordia Technologies: Smart Firewalls
  - UCLA/UCB/Utah (DARPA ANETS): Adaptive multimedia



Conclusions & Future Work

- What will it take to create autonomic systems?
  - Standardization of instrumentation technologies
  - Analyzable change propagation
  - New operational procedures
- Thesis contribution
  - Cut-through approach to the technology issues to prove feasibility
- Future work
  - Scaling development of change propagation models
  - Handling the dynamics of change propagation
  - Managing the autonomic management layer