Rainbow: Engineering Support for Self-Healing Systems

CompArch 2009

David Garlan
June 2009

Carnegie Mellon
institute for SOFTWARE RESEARCH

Joint work with …

- Owen Cheng
  - PhD Thesis May 2008, now at NASA JPL
- Bradley Schmerl
  - Staff
- Jung Soo Kim
  - Current PhD student

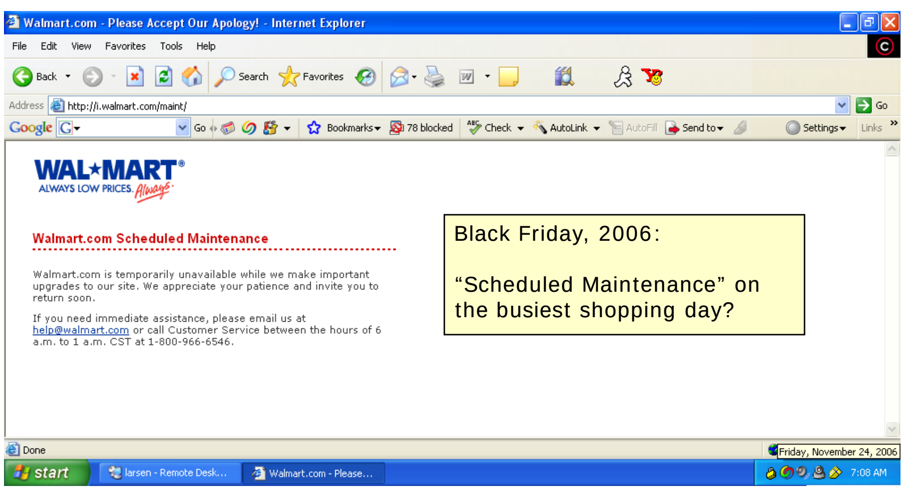
© David Garlan 2009

The Problem

- An important requirement for modern software-based systems

Maintain high-availability and optimal performance even in the presence of
- changes in environment
- system faults
- changes in user needs and context

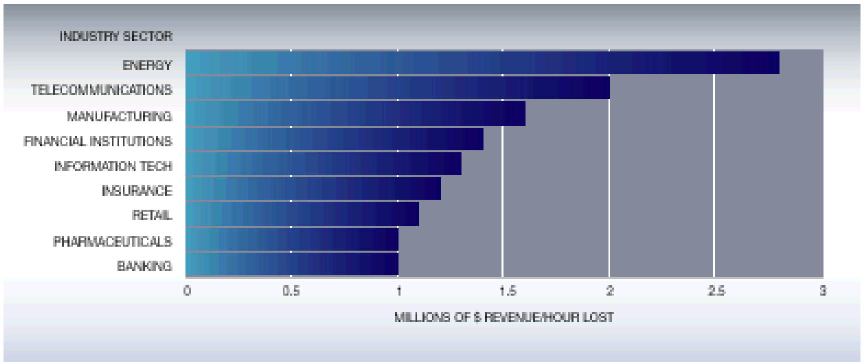
Websites Fail to Adapt



- Amazon.com disrupted due to Xbox 360 on the day before

Cost of Downtime

- Average hourly impact of downtime by industry sector



Data from *IT Performance Engineering and Measurement Strategies: Quantifying Performance Loss*, Meta Group, Stamford, CT (October 2000).

How is this addressed today?

- Technique 1: try to build resilience into application code directly
  - Use exceptions, timeouts, and other low-level programming mechanisms
- Unfortunately, this approach is not good for
  - Handling changing objectives
  - Locating the true cause of the problem
  - Detecting “softer” system anomalies
  - Maintainability: hard to add and modify adaptation policies and mechanisms
  - Legacy systems: hard to retrofit later

How is this addressed today?

- Technique 2: human oversight
  - Operators, system administrators
  - Global oversight, intelligent response
- Unfortunately, this approach is
  - Costly
  - Error-prone

© David Garlan 2009

Cost of Human Oversight

- Estimated 1/3-1/2 total IT budget to prevent or recover from crash
- "For every dollar to purchase storage, you spend $9 to have someone manage it"— Nick Tabellion
- Administrative cost: 60-75% overall cost of database ownership
- 40% of root causes of computer system outage is attributable to operator error

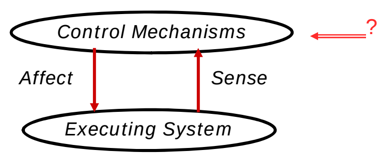
© David Garlan 2009

A New Approach

- Goal: systems automatically and optimally adapt to handle
  - changes in user needs
  - variable resources
  - faults

But how?

Answer: Move from open-loop to closed-loop systems



Model-based Adaptation

© David Garlan 2009

Example: Google File System

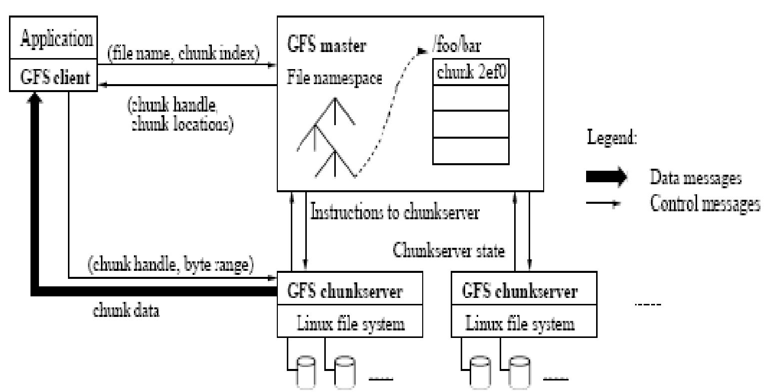


Figure 1: GFS Architecture

Source: "The Google File System" Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. SOSP 2003.

© David Garlan 2009

The Challenge

- Engineer self-adaptation to support
  - Cost-effectiveness
  - Legacy systems
  - Domain-specific adaptations
  - Multiple quality dimensions
  - Ease of changing adaptation policies

© David Garlan 2009

This Talk

- *Software Architectures* for dynamic self-adaptation
  - *Rainbow*: a framework in which architectural models can be used to adapt systems
  - *Stitch*: a new language to express self-adaptation strategies
- Examples and evaluation
- On-going work
- Related work

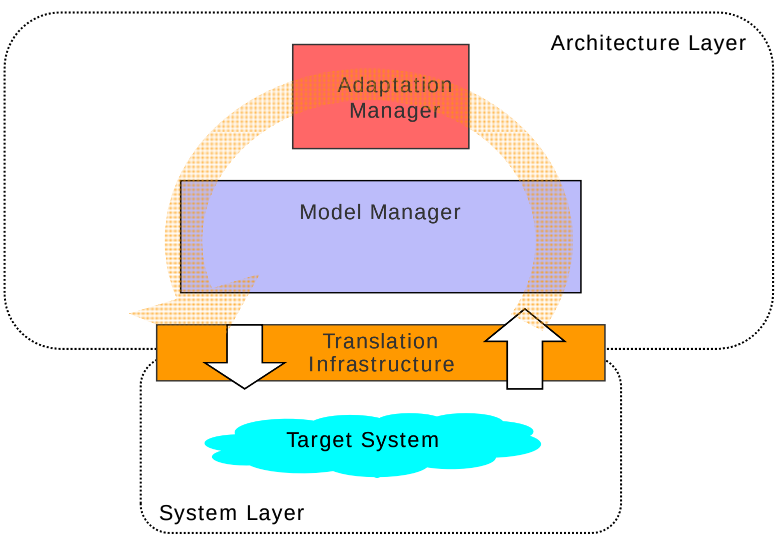
© David Garlan 2009

Our Approach

- **Framework**
  - Called Rainbow
  - Common infrastructure with reusable mechanisms
  - Customization points for tailoring to the domain
- **Language**
  - Called Stitch
  - Expresses adaptation policies as *strategies*
  - Expresses business quality dimensions and preferences
- Adaptation engineering **process**
  - Process for customizing the Rainbow framework
  - Roles: for engineering self-adaptation capabilities

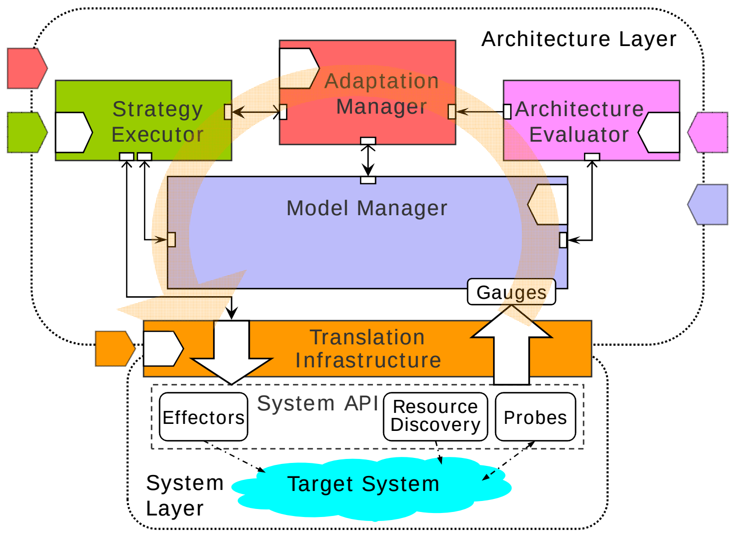
© David Garlan 2009

The Rainbow Framework

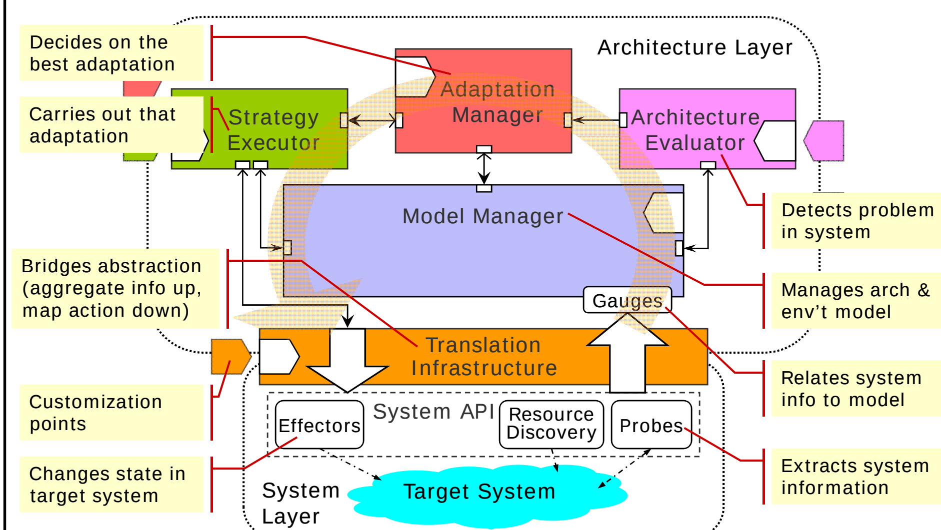


© David Garlan 2009

Rainbow Framework Overview

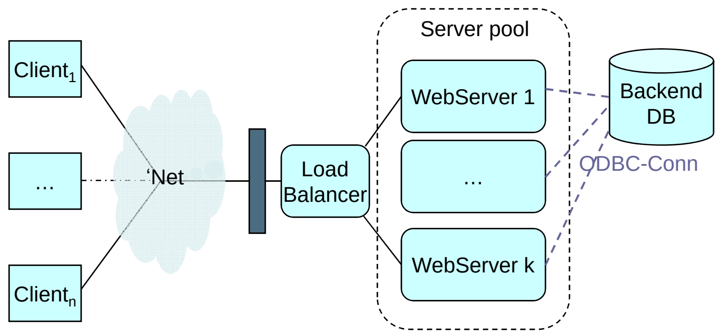


Rainbow Framework Overview



Self-Adaptation Example: *Znn.com*

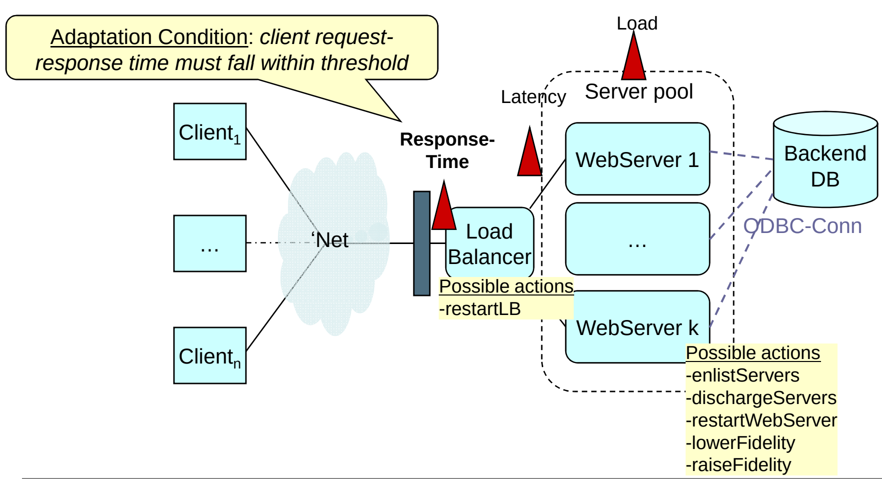
- We'll use this example throughout our talk



© David Garlan 2009

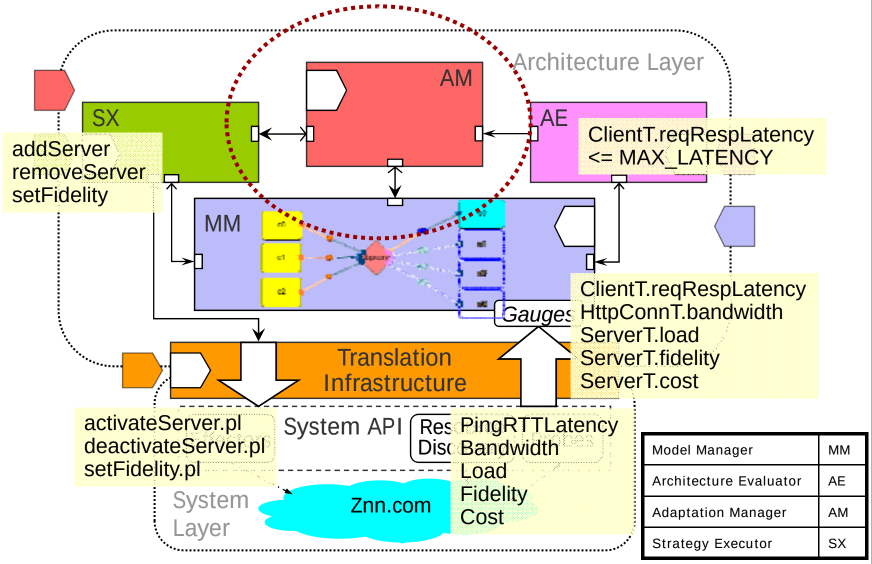
Self-Adaptation Example: *Znn.com*

- We'll use this example throughout our talk



© David Garlan 2009

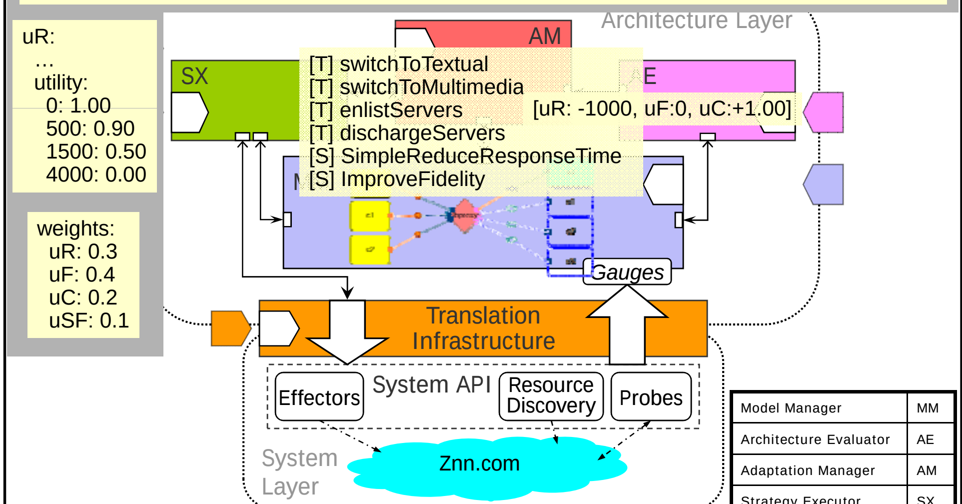
Znn.com: Rainbow Customizations (Part 1)



Model Manager	MM
Architecture Evaluator	AE
Adaptation Manager	AM
Strategy Executor	SX

Znn.com: Rainbow Customizations (Part 2)

Objectives: timely response (uR), high-quality content (uF), low-provision cost (uC)



Model Manager	MM
Architecture Evaluator	AE
Adaptation Manager	AM
Strategy Executor	SX

Stitch Language Overview

- Language requirements
  - **Concepts**: concepts to formalize operational aspects of self-adaptation
  - **Value system**: way to specify value system for comparing adaptations
  - **Choice**: apply value system to select best course of adaptation
- **Concepts** to express adaptation knowledge
  - *Operator* – basic system-provided command
  - *Tactic* – action with impact on quality dimensions
  - *Strategy* – adaptation with intermediate steps of condition-action-delay
- A way to specify **value system**
  - *Quality dimensions* – business QoS concerns
  - *Utility preferences* – business priorities over dimensions
  - *Adaptation conditions* – opportunities for improvement
- **Choice** of best adaptation
  - Strategy selection – best adaptation for current system conditions

© David Garlan 2009

Innovative features of Stitch

- Control-system model: Selection of next action in a strategy depends on observed effects of previous action
- Value system: Utility-based selection of best strategy allows context-sensitive adaptation
- Asynchrony: Timing delays capture settling time
- Uncertainty: effectiveness of a given tactic is known only within some probability

© David Garlan 2009

Stitch Language Constructs

Constructs

- **Operator**
- Tactic
- Strategy
- Quality dimensions
- Utility preferences
- Adaptation conditions

- A basic command provided by target system
  - Defined in the architectural style
  - Mapped to system-level effectors

addServer
removeServer
setFidelity

↓

activateServer.pl
deactivateServer.pl
setFidelity.pl

Stitch Language Constructs

Constructs

- Operator
- **Tactic**
- Strategy
- Quality dimensions
- Utility preferences
- Adaptation conditions

- Provides an **abstraction** to
  - **Package** operators into larger units
  - Form primitive adaptation **step**
  - Associate **cost-benefit** impact on quality dimensions
- Characteristics
  - An action composed of a sequence of operators
  - Guard conditions to determine applicability
  - Expected effects to observe in target system
  - Specified impacts on quality dimensions
  - Execution context: architectural style (types + operators) and model instance
  - Read-only snapshot of model during execution
  - Uses operators, provides primitives for *strategies*
- Cost-benefit attributes
  - $[a_1{:}\Delta v_1, \ldots, a_n{:}\Delta v_n]$ [uR: -1000, uF:0, uC:+1.00]

Stitch Language Constructs

Constructs
- Operator
- **Tactic** (example)
- Strategy

```
module newssite.tactics.example;
import model "ZnnSys.acme" { ZnnSys as M, ZnnFam as T };
import op "newssite.operator.ArchOp" { ArchOp as Sys };

tactic switchToTextualMode () {
  condition {
    exists c : T.ClientT in M.components | c.experRespTime >
     M.MAX_RESPTIME;
  }
  action {
    svrs = { select s : T.ServerT | !s.isTextualMode };
    for (T.ServerT s : svrs) {
      Sys.setTextualMode(s, true);
    }
  }
  effect {
    forall c : T.ClientT in M.components | c.experRespTime <=
     M.MAX_RESPTIME;
    forall s : T.ServerT in M.components | s.isTextualMode;
  }
}
```

© David Garlan 25

Stitch Language Constructs

Constructs
- Operator
- **Tactic** (example)
- Strategy

```
module newssite.tactics.example;
import model "ZnnSys.acme" { ZnnSys as M, ZnnFam as T };
import op "newssite.operator.ArchOp" { ArchOp as Sys };

tactic switchToTextualMode () {
  condition {
    exists c : T.ClientT in M.components | c.experRespTime >
     M.MAX_RESPTIME;
  }
  action {
    svrs = { select s : T.ServerT | !s.isTextualMode };
    for (T.ServerT s : svrs) {
      Sys.setTextualMode(s, true);
    }
  }
  effect {
    forall c : T.ClientT in M.components | c.experRespTime <=
     M.MAX_RESPTIME;
    forall s : T.ServerT in M.components | s.isTextualMode;
  }
}
```

Import of model and operators

Conditions of applicability

Sequence of operators

Expected effects to observe

© David Garlan 26

Stitch Language Constructs

Constructs
- Operator
- Tactic
- **Strategy**
- Quality dimensions
- Utility preferences
- Adaptation conditions

- Encapsulates path of adaptations where
  - Each step is conditional execution of a tactic
  - Choice of tactic depend on intermediate results
    - Intermediate step might reduce utility
  - Timing delay for asynchrony in achieving effect
  - Probabilities on conditions to handle inherent uncertainties in achieving effect
- Characteristics
  - Tree of condition-action-delay decision nodes
  - Conditions of applicability to determine involvement
  - Execution context: arch style (types) and tactics
  - Intermediate system observations at decision nodes
  - Enables computing aggregate cost-benefit attributes for utility-based strategy selection

© David Garlan

Stitch Language Constructs

Constructs
- Operator
- Tactic
- **Strategy** (e.g.)

```
module newssite.strategies.example;
import model "ZnnSys.acme" { ZnnSys as M, ZnnFam as T };
import lib "newssite.tactics.example";
import op "org.sa.rainbow.stitch.lib.*";  // Model, Set, & Util

define boolean styleApplies = ... Model.hasType(M, "ServerT");
define boolean cViolation = exists c : T.ClientT in M.components |
    c.experRespTime > M.MAX_RESPTIME;

strategy SimpleReduceResponseTime [ styleApplies && cViolation ] {
  define boolean hiLatency = exists conn : T.HttpConnT in
    M.connectors | conn.latency > M.MAX_LATENCY;
  define boolean hiLoad = exists s : T.ServerT in M.components |
    s.load > M.MAX_UTIL;

  t1: (#[Pr{t1}] hiLatency) -> switchToTextualMode() @[1000/*ms*/] {
    t1a: (success) -> done ;
  }
  t2: (#[Pr{t2}] hiLoad) -> enlistServer(1) @[2000/*ms*/] {
    t2a: (!hiLoad) -> done ;
    t2b: (!success) -> do [1] t1 ;
  }
  t3: (default) -> fail;
}
```

© David Garlan

Stitch Language Constructs

Constructs
- Operator
- Tactic
- **Strategy** (e.g.)

Import of model and tactics

Condition of applicability

Step of condition-action-delay

Timing delay

Tactic succeeded?

Complete strategy

No branch matches?

Abort strategy

```
module newssite.strategies.example;
import model "ZnnSys.acme" { ZnnSys as M, ZnnFam as T };
import lib "newssite.tactics.example";
import op "org.sa.rainbow.stitch.lib.*";  // Model, Set, & Util

define boolean styleApplies = … Model.hasType(M, "ServerT");
define boolean cViolation = exists c : T.ClientT in M.components |
    c.experRespTime > M.MAX_RESPTIME;

strategy SimpleReduceResponseTime [ styleApplies && cViolation ] {
  define boolean hiLatency = exists conn : T.HttpConnT in
    M.connectors | conn.latency > M.MAX_LATENCY;
  define boolean hiLoad = exists s : T.ServerT in M.components |
    s.load > M.MAX_UTIL;

  t1: (#[Pr{t1}] hiLatency) -> switchToTextualMode() @[1000/*ms*/] {
    t1a: (success) -> done ;
  }
  t2: (#[Pr{t2}] hiLoad) -> enlistServer(1) @[2000/*ms*/] {
    t2a: (!hiLoad) -> done ;
    t2b: (!success) -> do [1] t1 ;
  }
  t3: (default) -> fail;
}
```


Stitch Language Constructs

Constructs
- Operator
- Tactic
- Strategy
- **Quality dimensions**
- Utility preferences
- Adaptation conditions

- Intuitively, determine *what* to adapt for
- Notion of utility for particular values of attribute
- Profile: ID, label, description, mapping, function
  - Mapping: monitored architectural property → current system condition
  - Function: linear, sigmoid, custom (interpolated)

```
uR:
  label: "Average Response Time"
  mapping: "[EAvg] ClientT.experRespTime"
  description: …
  utility:
    0: 1.00
    500: 0.90
    1500: 0.50
    4000: 0.00
```

Stitch Language Constructs

Constructs
- Operator
- Tactic
- Strategy
- Quality dimensions
- **Utility preferences**
- Adaptation conditions

- Intuitively, determine *how* to adapt
- Resource constraints: can't optimize all qualities
- Technique: assign weights for relative importance
  - Weights must sum to 1

```
weights:
  uR: 0.3
  uF: 0.4
  uC: 0.2
  uSF: 0.1
```


Stitch Language Constructs

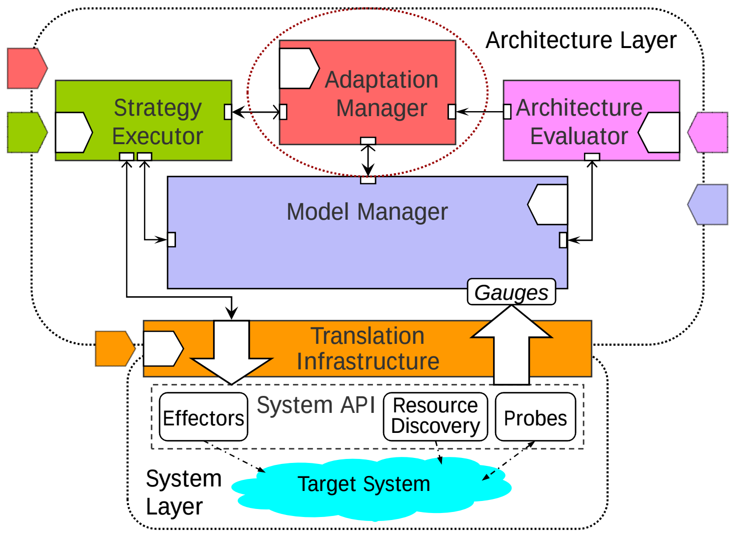
Constructs
- Operator
- Tactic
- Strategy
- Quality dimensions
- Utility preferences
- **Adaptation conditions**

- Intuitively, determine *when* to adapt
- Constraints defined in architectural style
- QoS threshold quantity defined in model instance

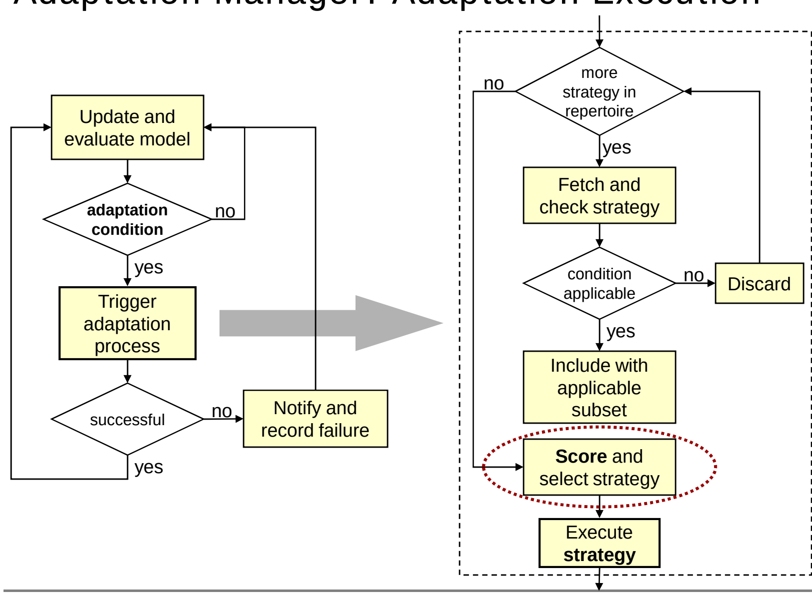
Invariant self.avg_latency < MAX_RESPTIME;

Property MAX_RESPTIME : **float** = 1000.0;

Rainbow Context for Adaptation Execution



Adaptation Manager: Adaptation Execution



Strategy Selection

- Given:
  - Quality dimensions and weights (e.g., 4)
  - A strategy with
    - N nodes
    - Branch probabilities as shown
    - Tactic cost-benefit attributes

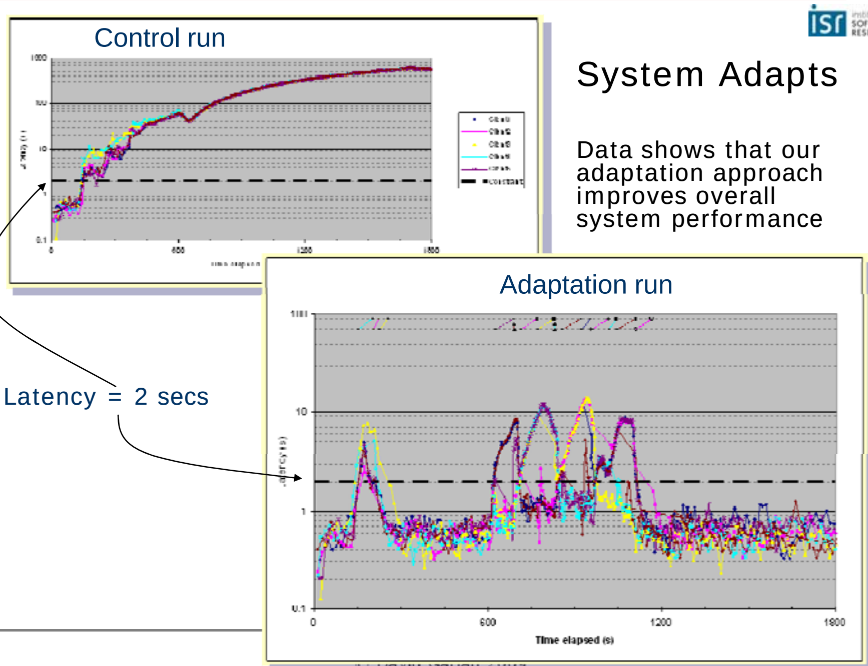
$u_{latency}()$, $u_{quality}()$, $u_{cost}()$, $u_{disruption}()$
$(w_{latency}, w_{quality}, w_{cost}, w_{disruption})$
$= (0.5, 0.3, 0.1, 0.1)$ **[= 1]**

<u>Algorithm</u>
Given tree *g* with node *x* and its children *c*:
EAAV (*g*) = *sysAV* + *AggAV* (*root* (*g*))
AggAV (*x*) = *cbav*(*x*) + Σ_c *prob*(*x*,*c*) *AggAV*(*c*)

- Propagate cost-benefit vectors up the tree, reduced by branch probabilities
- Merge expected vector with current conditions (assume: [1125, 3.5, 0, 0])
- Evaluate quality attributes against utility functions
- Compute weighted sum to get utility **score**

System Adapts

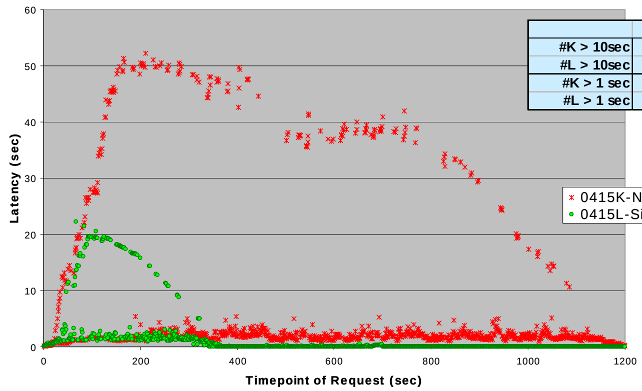
Data shows that our adaptation approach improves overall system performance



Latency = 2 secs

Znn.com Experiment Run Results

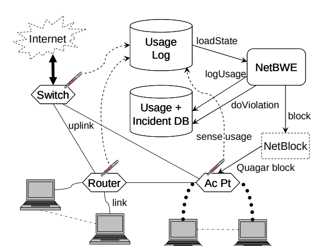
Aggregate JMeter Data										
		ms response								
URL	#Samples	Average	Median	90% Line	Min	Max	Error%	Throughput	KB/sec
0415K No Rainbow	1200	7981	1953	37514	93	52201	0.00%	8.4/sec	6.92
0415L SimpleReduce1	1200	1502	16	2187	0	22202	0.17%	29.5/sec	78.26



			% of K
#K > 10sec	217		
#L > 10sec	80		36.9%
#K > 1 sec		1120	
#L > 1 sec		298	26.6%

System Administration Examples

- Sys-admin interview
  - **Results**: Stitch concepts seem **natural** fit for sys-admin routines
  - Methodology: priming, interview, compose Stitch script from scenarios
  - White problem scenarios: scripts represented in Stitch
  - Almossawi problem scenarios: structure matches Stitch strategies



- Analysis using CMU sys-admin example: *Netbwe*
  - **Results**:
    - Rainbow captured adaptation concerns
    - Stitch **hoisted** policies buried in Perl code
  - Distinguishable adaptation tasks
    - Core commands as operators
    - Coarser-grained sequence of commands (step) with conditions of applicability and intended effects
    - Adaptations with intermediate condition-actions and observations

Related work

- Many names for this area of research
  - self-healing systems, autonomic computing, self-* systems
  - Hot area: conferences, workshops, journals
- Most existing approaches focusing on specific domains and quality attributes
  - Usually assumes particular architectural style
- Other techniques used
  - Planning, expert systems
- Traditional areas
  - Control systems, utility theory, Markov decision processes, etc.

© David Garlan 2009

On-going work

- Cyber-physical systems
  - Large-scale distributed control systems
  - Examples: power grid, air-traffic control, multi-car coordination, smart home
- Analysis
  - How can you decide if a given adaptation strategy is "sound"?
  - Can model-checking be applied to Stitch?
- Service-oriented architectures
  - Adapting service orchestrations to match environment
- Incorporating learning
  - To improve strategy selection
  - To learn new strategies

© David Garlan 2009

Conclusion

- Today systems **fail to adapt** to changing environments
- **Architecture models** and an **adaptation language** can be combined to self-adapt systems for multiple quality dimensions with broad scope
- **Rainbow**
  - integrates architecture model and a language for self-adaptation
  - provides software engineers the ability to add and evolve self-adaptation capabilities
    - *cost-effectively*
    - *for multiple objectives*
    - *with explicit customization points to tailor self-adaptation capabilities for particular classes of systems and quality objectives.*

"Somewhere Over the Rainbow, A Stitch in Time Saves Nine"

© David Garlan 2009