
Evaluating the Effectiveness of the Rainbow Self-Adaptive System

Shang-Wen Cheng
David Garlan
Bradley Schmerl

Software Engineering for Adaptive and
self-Managing Systems @ 31st ICSE

2009.05.18-19



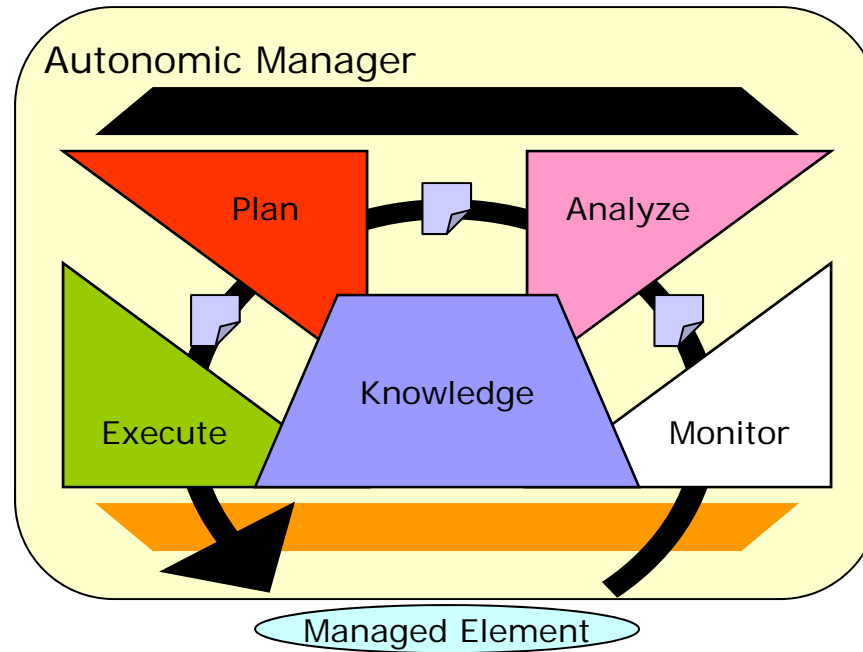
Outline

- Rainbow evaluation with Znn.com
 - Effective at maintaining QoS under changing conditions
 - Self-adaptive in a timely fashion
 - Low engineering effort required

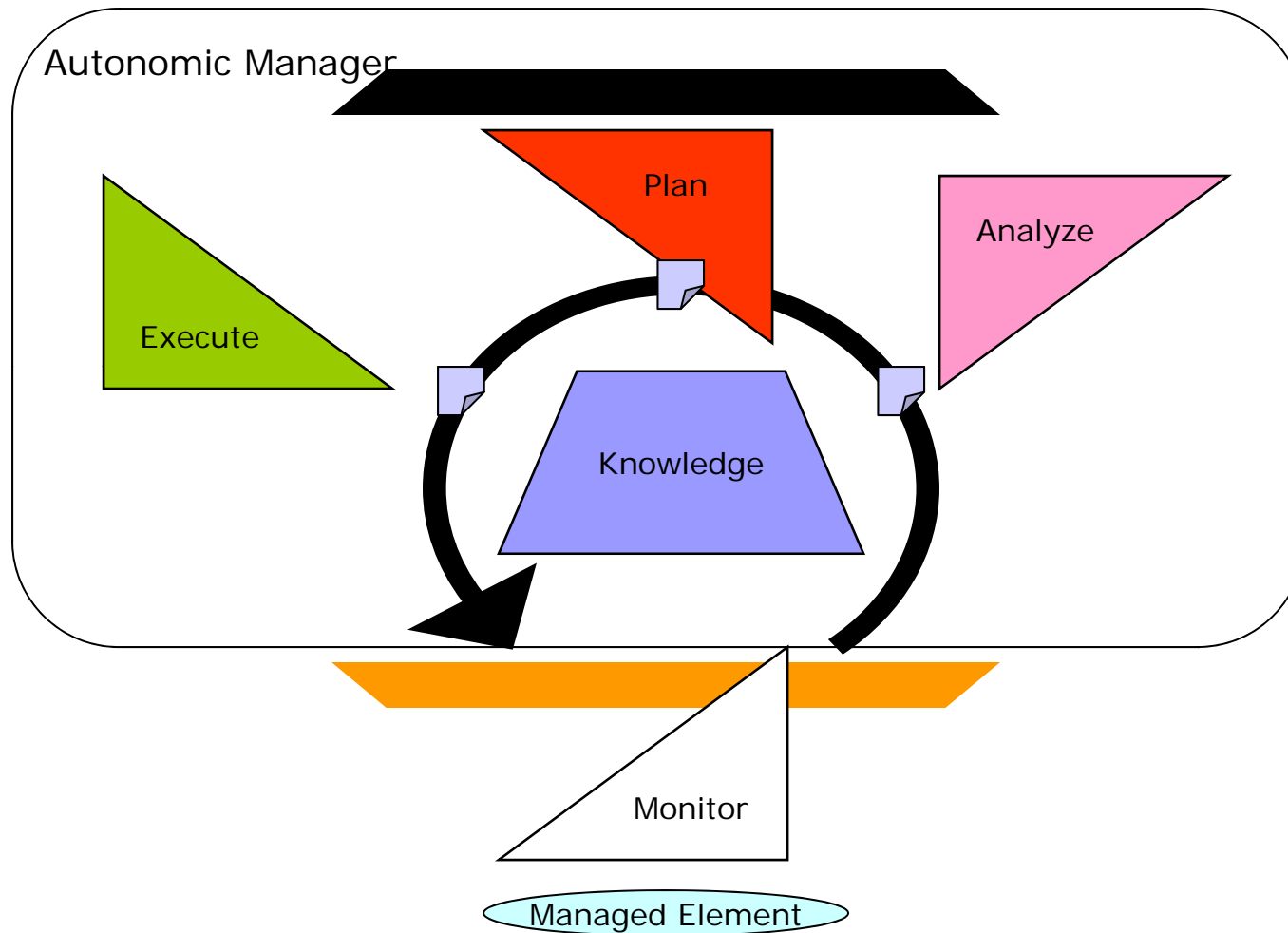
- Common benchmark for self-adaptive systems
 - Principles for benchmarking
 - SAFU: metric for comparison
 - Znn.com suite
(accessible to community)



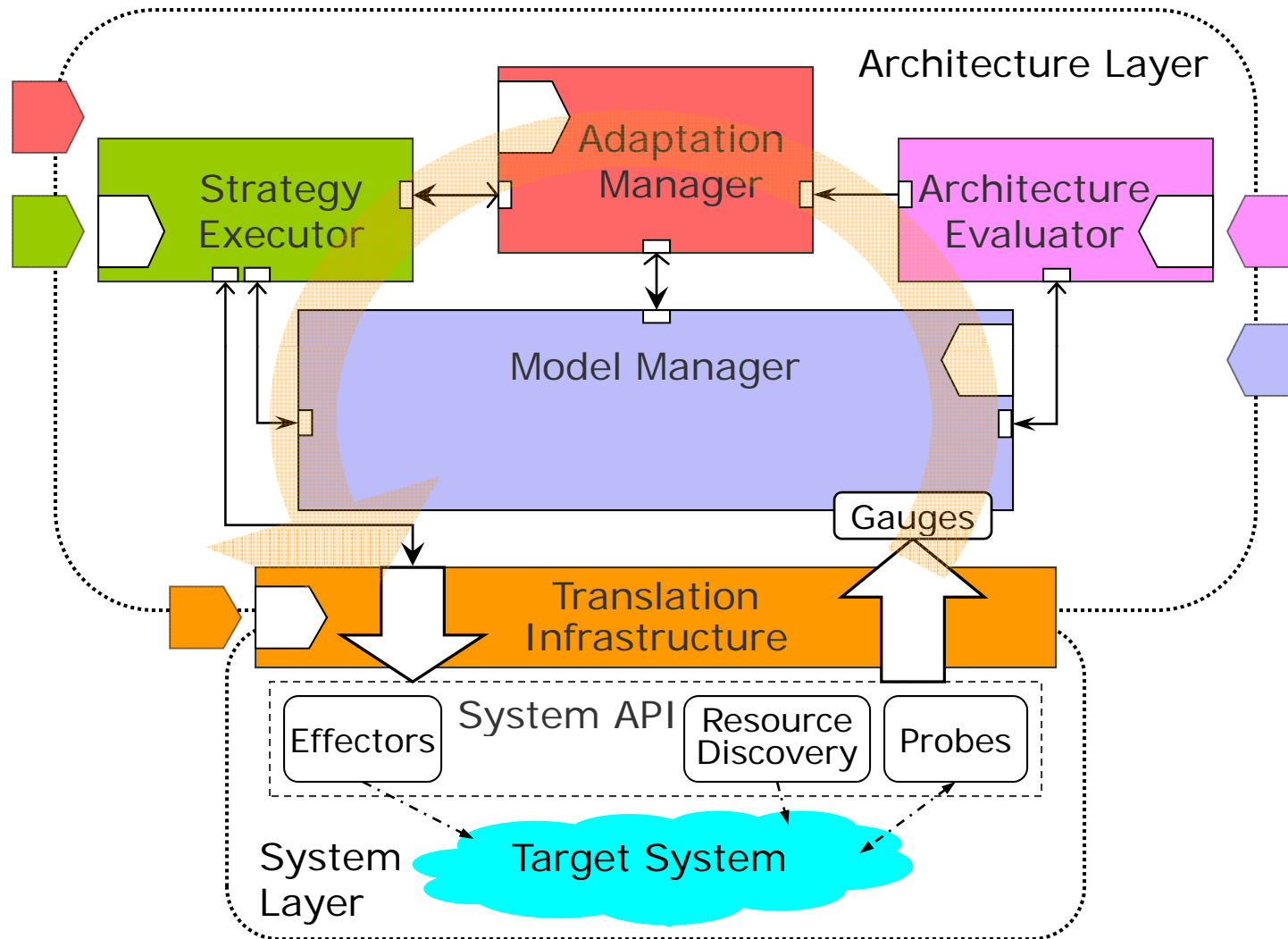
Brief Rainbow overview: a MAPE loop



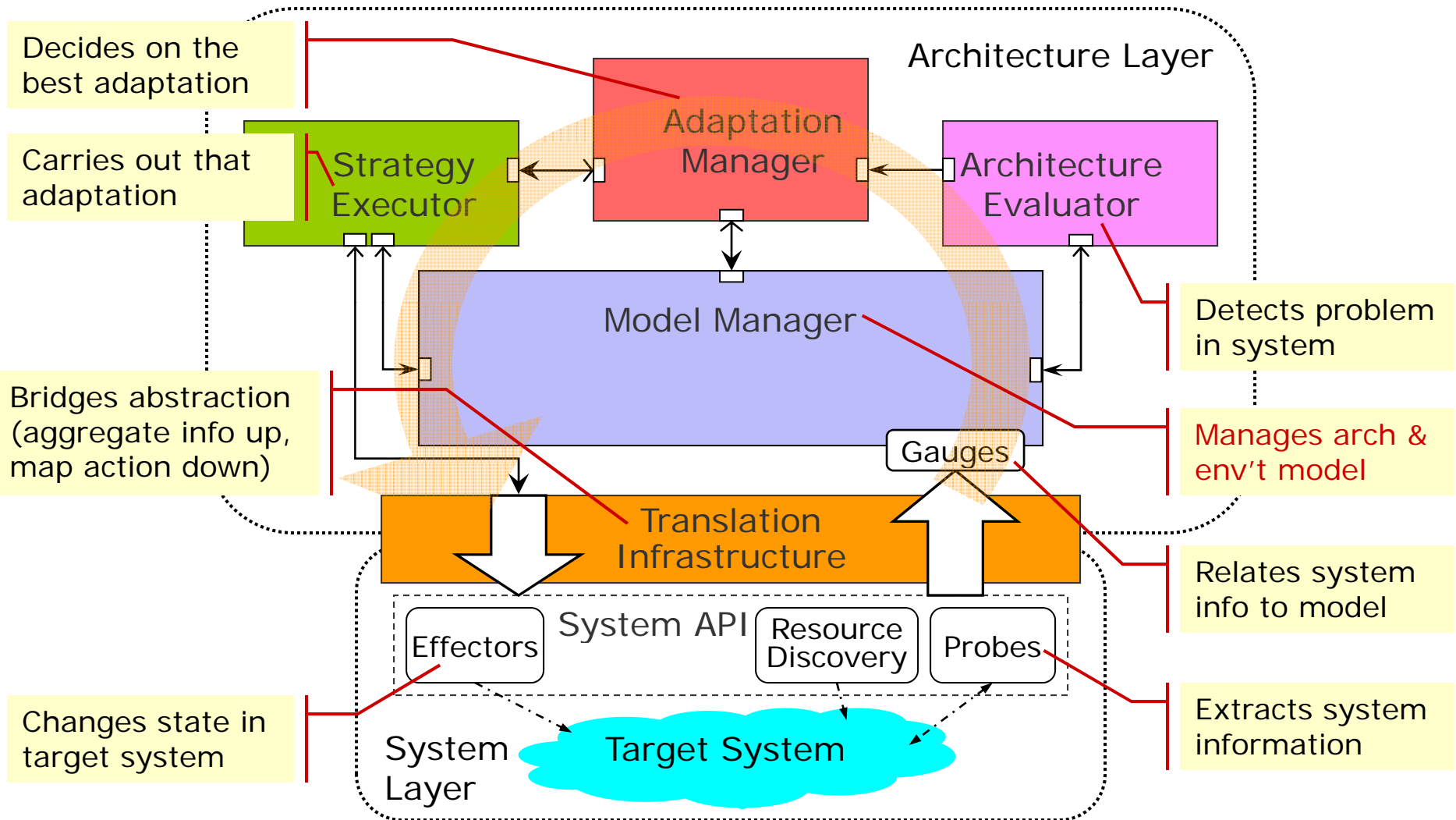
Brief Rainbow overview: a MAPE loop



Brief Rainbow overview: a MAPE loop



Rainbow framework overview



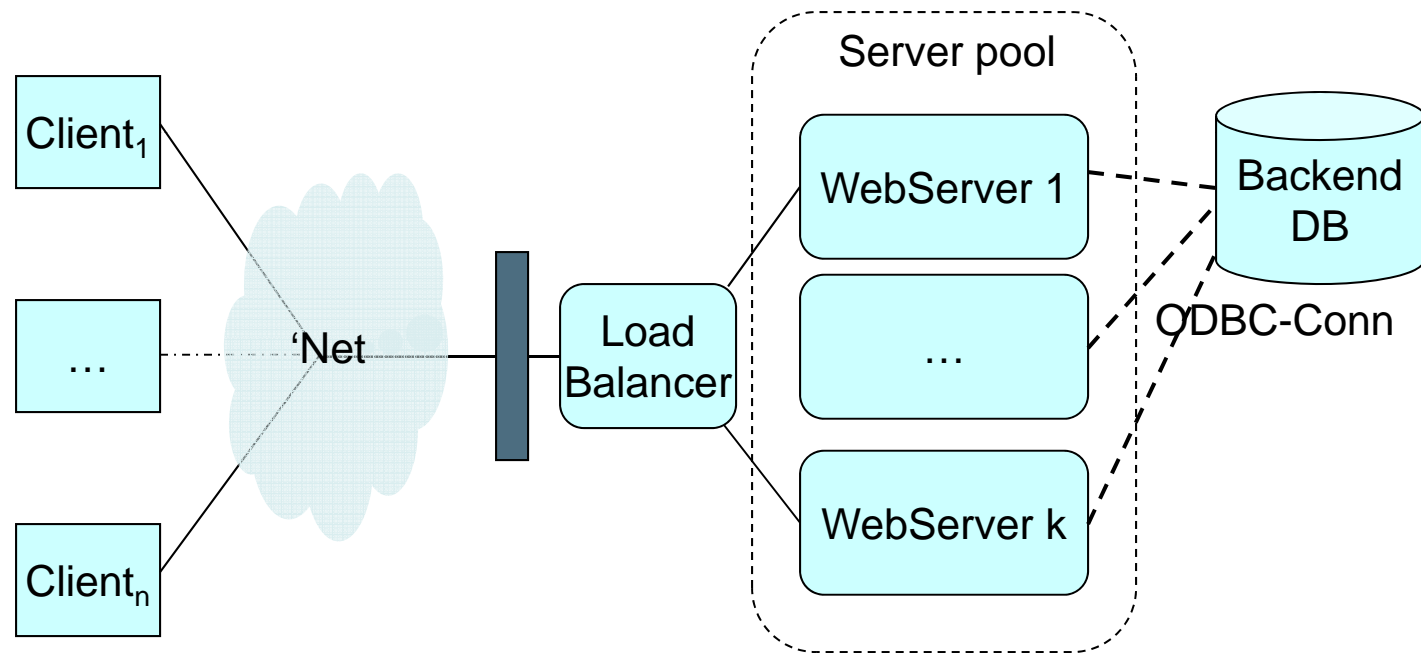
Evaluation criteria: utility, effort, generality

- How well does resulting system self-adapt?
 - Does it actually detect problems and fix them?
 - Does it improve **overall utility**?
 - How much run-time **resource overhead** does it incur
 - How much time does adaptation take?
 - Etc.
- How much **effort** is required to tailor Rainbow to target system?
- How many different styles of system and quality dimensions can Rainbow address?

Znn.com: a web-based client-server system

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

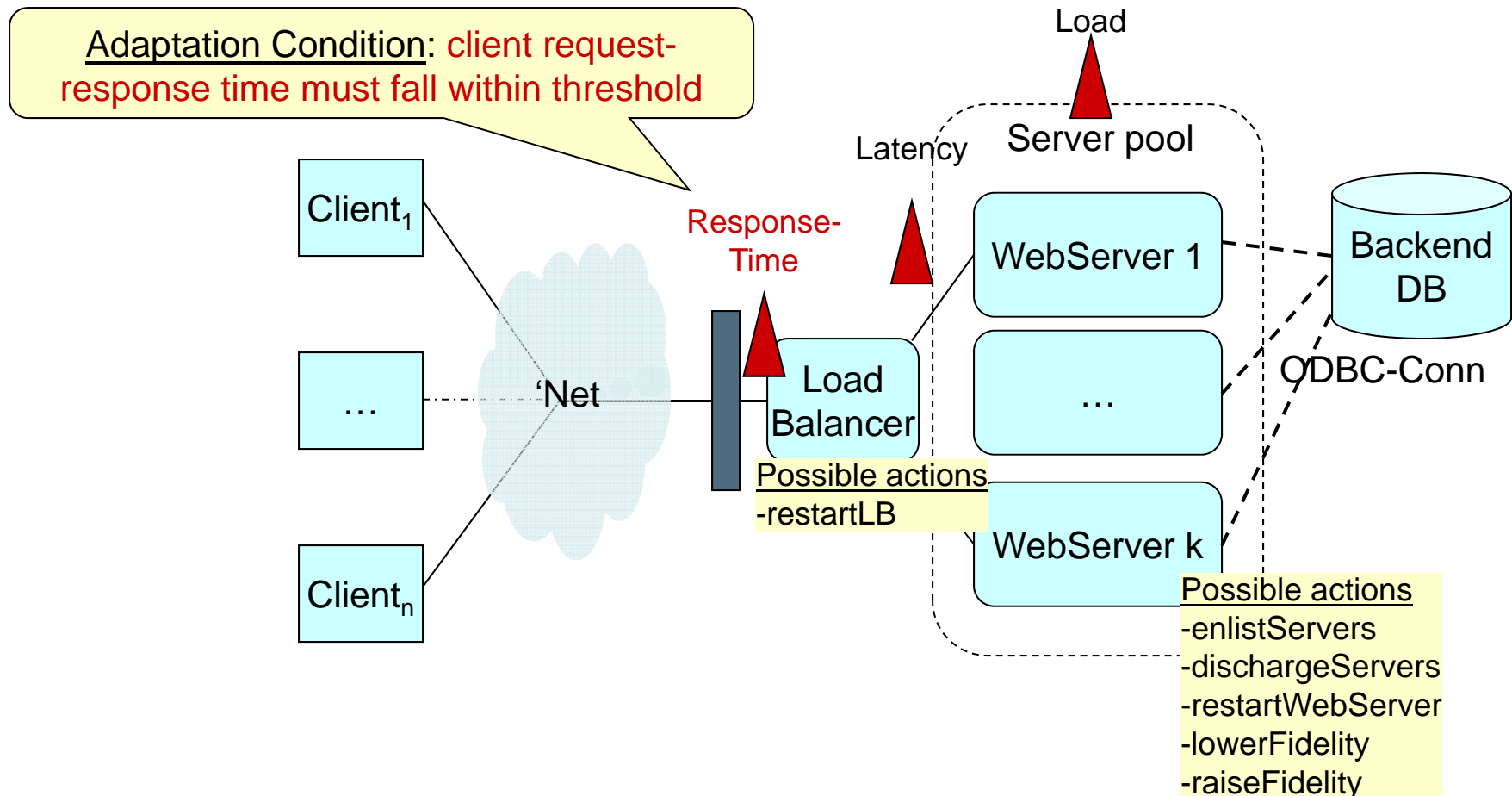
- 3-tiered variant of web-based client-server system



Znn.com: a web-based client-server system

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

- 3-tiered variant of web-based client-server system



General benchmark systems

- **Relevance** to real-world problems
- **Accessibility** of codebase, testable in *standalone* environment
- Can **observe** and **change** system
- **Metrics** for comparison, e.g.,
 - CPU: MIPs and transistor count
 - Algorithm: run time/input size
 - Databases: transactions/sec
- Ease of **extracting** performance data from system and environment for evaluation

Specific to self-adaptive systems

- **Versatility** for wide range of self-adaptive approaches and application domains
- Allow changing system **dynamically**
- Support multiple quality **dimensions**
 - Trade-off
 - E.g., availability, performance, reliability, security
- Allow multiple adaptive **operations**
- Provide multiple paths of **configuration** to achieve same goal

Znn.com: how to access and use

- Benchmark suites at <http://rainbow.self-adapt.org/benchmark>
- Required system and software
 - Any commodity server hardware
 - Apache HTTPD, PHP, MySQL, Perl
 - On manager/tester node: Cygwin (if Windows) & Jakarta JMeter
- How to use
 - Set up software components and Znn.com site
 - Configure system for your self-management framework
 - Your goals for adaptation
 - Your sensing/probing and actuating/effecting needs
 - Your paths of configuration
 - Bombard with JMeter (a traffic generation & measurement tool)
 - Extract data from JMeter log for analysis

Znn.com suitable as benchmark suite?

- **Relevance**: Slashdot effect, no perfect solution, several quality dimensions
- **Accessible** open-source parts: Apache Webserver, PHP, MySQL; Perl, JMeter
- Suite of probes (to **observe**) and effectors (to **change**)
 - CPU load, disk I/O rate, available bandwidth, ApacheTop, content fidelity
 - Start/stop server, alter content fidelity, randomly reject client requests
 - (Perl scripts vs. Apache modules)
- **Extract** performance data from target system for evaluation
 - Client-server style gauges: End2EndLatencyGauge, LoadGauge
 - Request traffic generation & measurement tool: JMeter
- Multiple points of variability
 - Quality **dimensions**: performance concerns, availability, reliability, security, and even data privacy
 - Adaptive **operations** and **configuration** paths
- Not met: **versatility** and **metric** requirements

SAFU as a self-adaptive system metric

- Self-Adaptation Fitness Unit
 - 100 indicates perfect (unattainable) achievement of all adaptation goals
- Comparison between two self-adaptation techniques comprises:
 - Their relative SAFU
 - How they contrast in dimensions and weights
- Shared profiles of SAFU
 - Common set of dimensions and weights for particular domains

Criteria	Value	Weight
Quality dimensions		40%
- Response time		
- Content fidelity		
- Provision cost		
Resource overhead		30%
Engineering effort		30%
Self-Adaptation Fitness Unit (SAFU):		

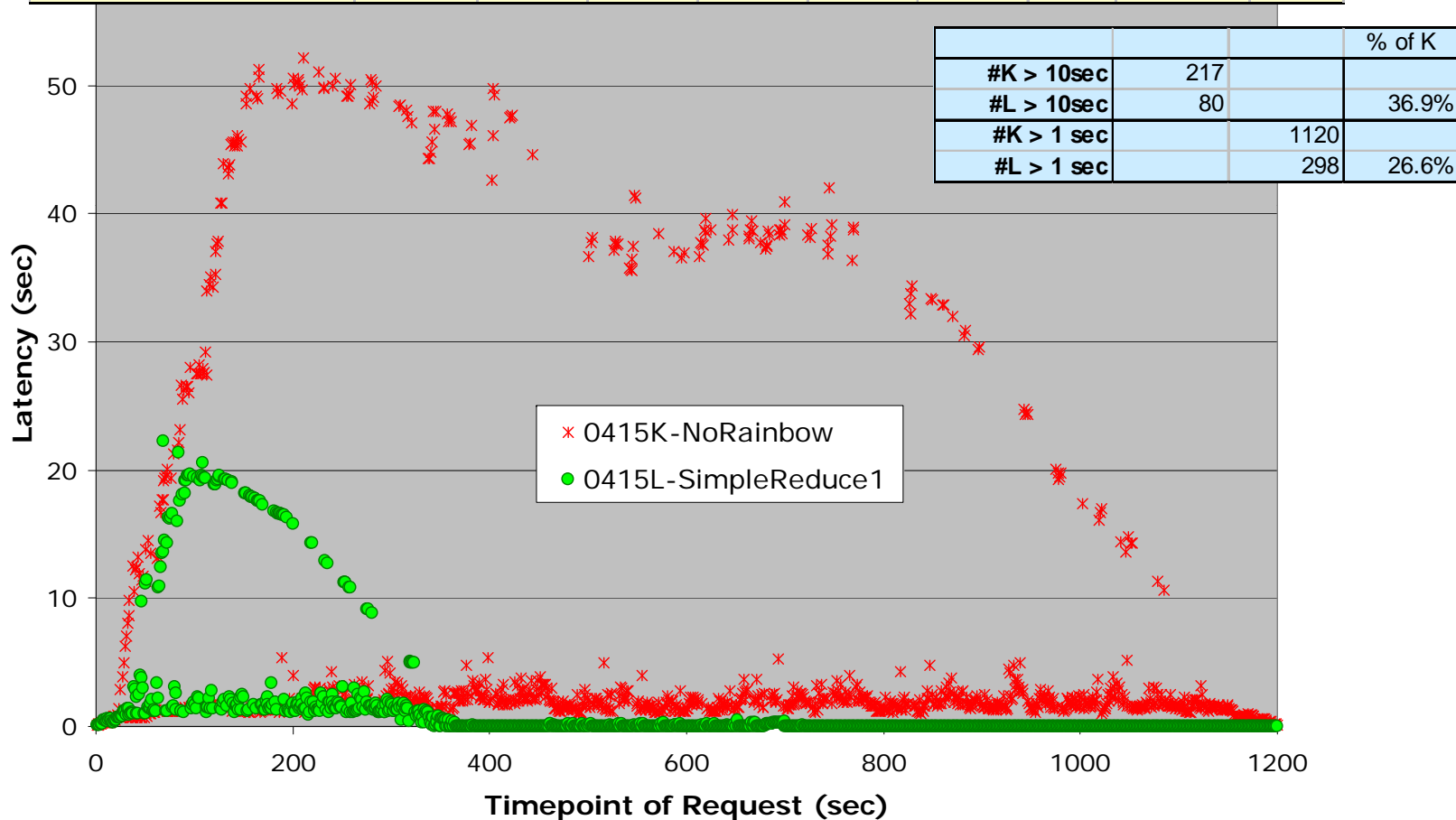
Effectiveness criteria: utility, overhead, effort

- Rainbow-ized target system self-adapts for improved overall utility
 - 1034 of 1200 possible units of accrued utility
- Self-adaptation incurs low run-time resource overhead, and
 - 2% of CPU (occasionally ~5-10%), ~2MB of memory footprint
 - Further reducible with implementation optimization
- Effort required to tailor Rainbow to the target system significantly lower than developing self-adaptive capabilities from scratch

		Duration	
No	Customization Task	Znn.com	TalkShoe
1	Target system monitors and effectors	56	13
2	Model capture	13	4
3	Stitch script	21	9
4	Roundtrip integration + modification	3	8
	Total customization time (man-hrs):	93 h	34 h (56)

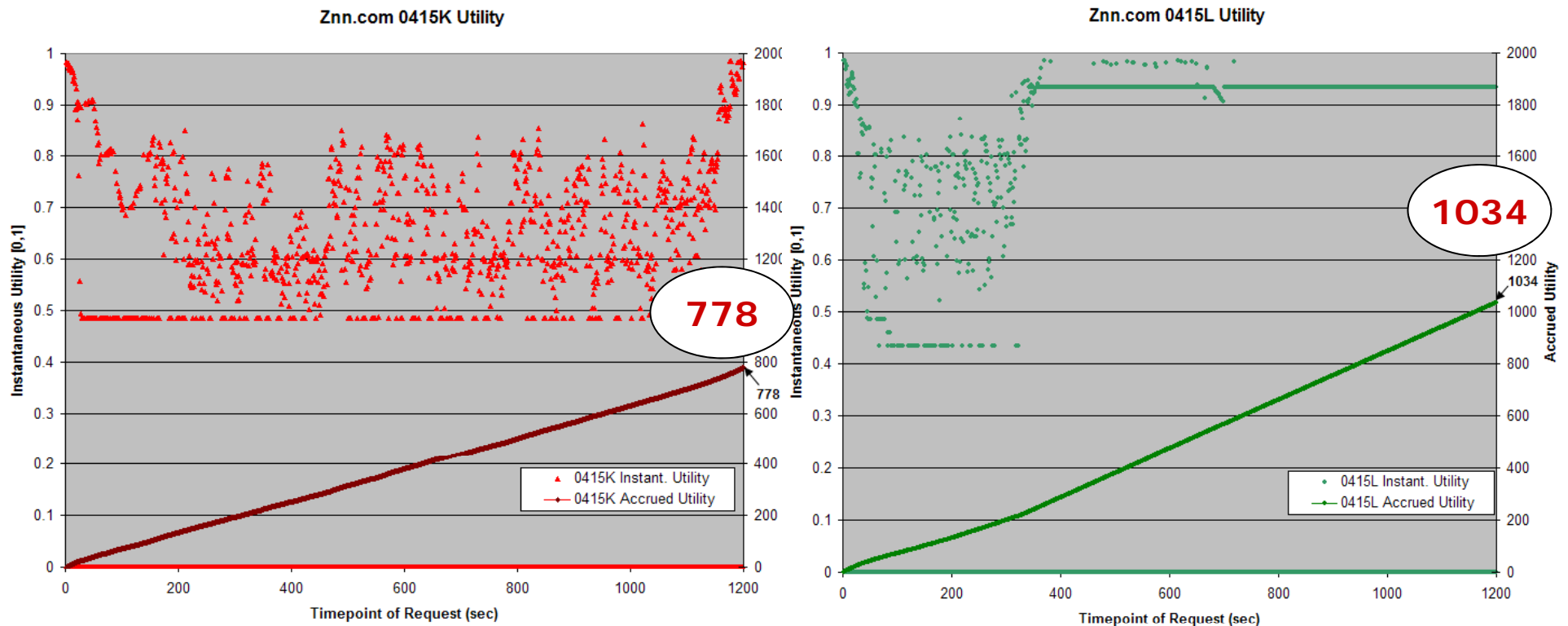
Rainbow-ized *Znn.com* experiment data

Aggregate JMeter Data									
URL	#Samples	ms response					Error%	Throughput	KB/sec
		Average	Median	90% Line	Min	Max			
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



Rainbow-ized *Znn.com* experiment data

- Data shows 33% improvement in Accrued Utility



SAFU a quantitative metric for comparison?

- Self-Adaptation Fitness Unit
 - 100 indicates perfect (unattainable) achievement of all adaptation goals
- Comparison between two self-adaptation techniques comprises:
 - Their relative SAFU
 - How they contrast in dimensions and weights
- Shared profiles of SAFU
 - Common set of dimensions and weights for particular domains

Criteria	Value	Weight
Quality dimensions	86 (1034/1200 AU)	40%
- Response time	994	50%
- Content fidelity	1021	25%
- Provision cost	1128	25%
Resource overhead	90-95 (5-10% overhead)	30%
Engineering effort	30-45 (days-weeks)	30%
Self-Adaptation Fitness Unit (SAFU):		70-76

$$86 \times 40\% + 95 \times 30\% + 45 \times 30\% = 76$$

Concluding and seeding discussion

- In summary,
 - Rainbow is effective in terms of utility, overhead, and effort
 - Znn.com qualifies as suitable benchmark suite
- Future work: generalize instance into a benchmark “framework”
 - Plug-in of capabilities for specific self-adaptation agenda
 - Web resources to collect and disseminate experience of use
- Community discussion: **what’s needed to compare approaches?**
 - **Engineering cost** (legacy systems) good criterion of comparison?
 - **SAFU** as a **common quantitative metric**?
 - What would be a **versatile** benchmark suite?
 - Specific benchmarks for mobility
 - Multiple environmental contexts (ubicomp)
 - Dynamic components (genetic programming)
 - Different self-* properties (self-organizing systems)

The End

- Evaluating the Effectiveness of the Rainbow Self-Adaptive System

Shang-Wen Cheng
David Garlan
Bradley Schmerl

Carnegie Mellon University School of Computer Science
{zensoul,garlan,schmerl}@cs.cmu.edu

- Benchmark suite at <http://rainbow.self-adapt.org/benchmark>
 - Also, SEAMS has a Wiki! <http://seams.self-adapt.org/>
 - Rainbow framework soon to be open source
- Questions?

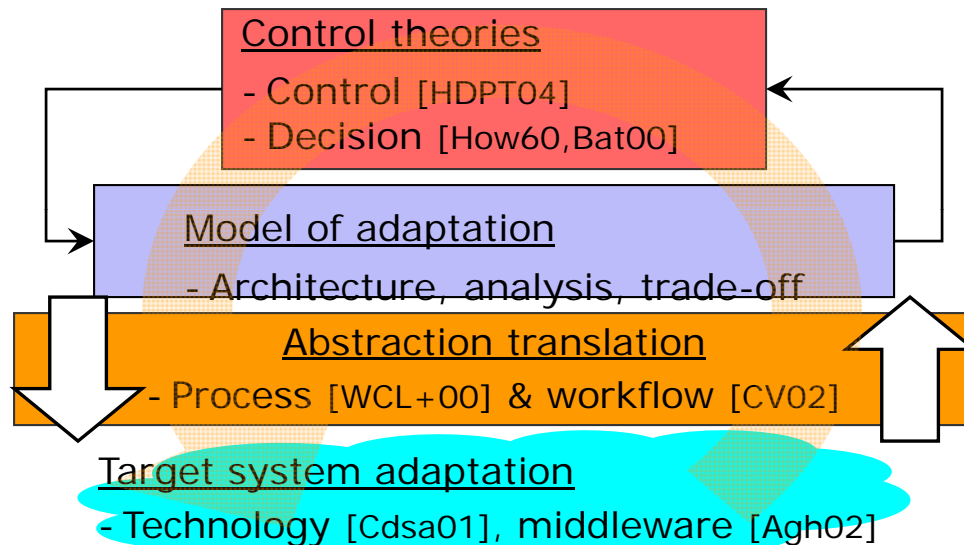
Supplemental Slides

Cheng, Garlan, Schmerl

- What's needed to compare approaches?
 - Engineering cost (legacy systems) good criterion of comparison?
 - SAFU as a common quantitative metric?
 - What would be a versatile benchmark suite ?
 - Specific benchmarks for mobility
 - Multiple environmental contexts (ubicomp)
 - Dynamic components (genetic programming)
 - Different self-* properties (self-organizing systems)
- Discussion at several levels
 - SAFU score vs. SAFU profiles
 - Znn.com example system
 - More cases: book store, airport services, traffic controll, etc.
 - Criteria of research, e.g., dependability, non-intrusiveness, etc

Related work

- Commonality:
 - Closed-loop control (MAPE)
 - Architecture model [OGT+99]



- Evaluation of autonomic systems:
 - Database optimization [MLR03]
 - Server provisioning [RMC03]
 - Workload optimization [SILM06]
 - Nine evaluation criteria [MH04]

Related approaches

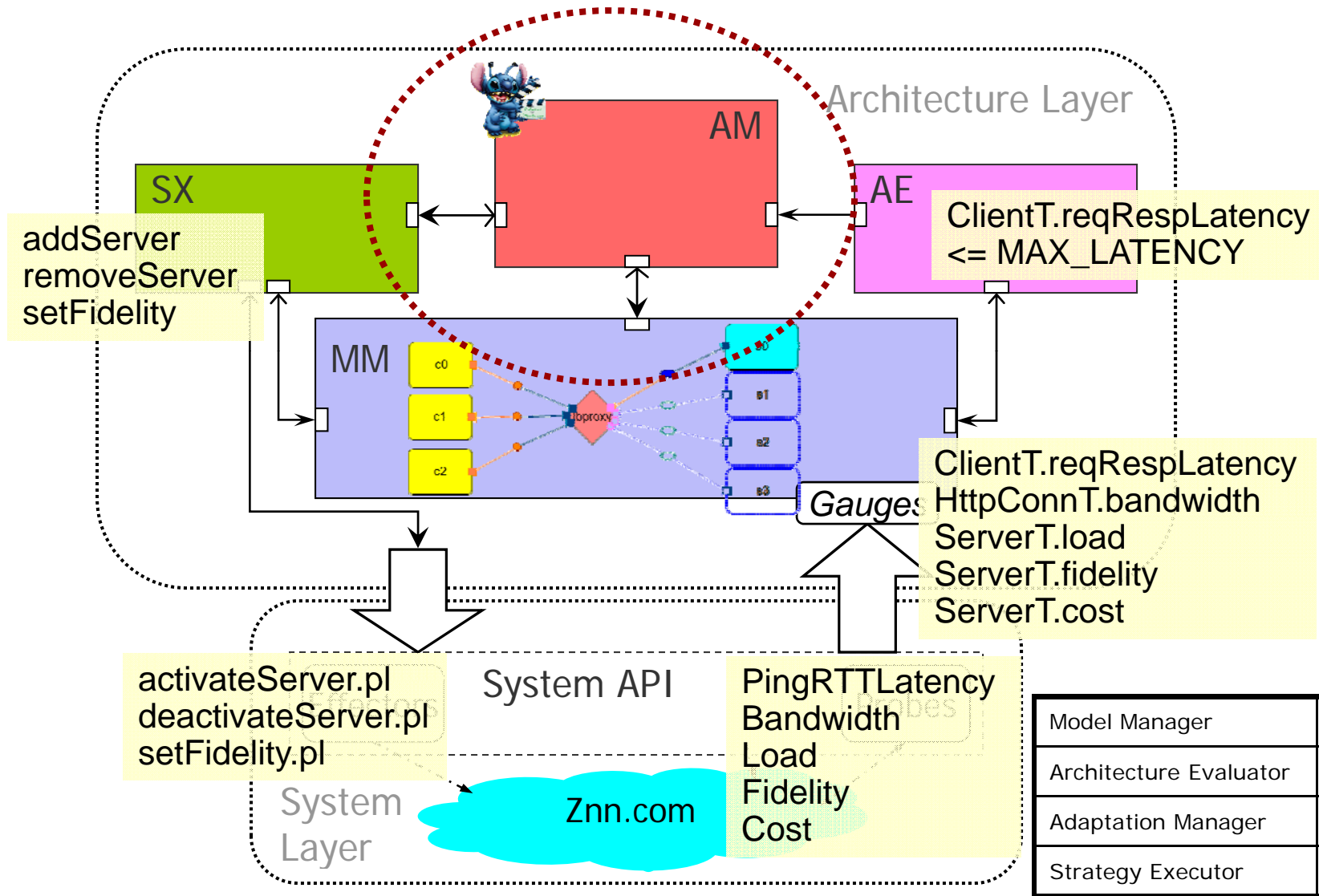
- IBM Autonomic Computing [GC03]
 - Specific IT challenges, e.g., log analysis, system deployment
- Architecture-based self-adaptation
 - Distributed, decentralized [GMK02]
- **Assumed system style**
 - Weaves: data-flow [GR91]
 - Arch Evolution Framework [DHT02]
 - Plastik: OpenCOM [BJC05]
 - Adaptive Server Framework [LG07]
 - Adaptive pipeline IS [HPUM07]
- **Particular quality dimensions**
 - Willow: survivability [WHC+01]
 - CASA: resource availability in mobile network [MG04]
 - CR-RIO: CBabel + contracts + profile + reconfiguration [SL06]

Rainbow: **generic** to **styles** and handles **multiple objectives**

McCann's 9 evaluation criteria

- Quality of Service
- Cost
- Granularity/Flexibility
- Failure avoidance (Robustness)
- Degree of Autonomy
- Adaptivity
- Time to adapt and Reaction Time
- Sensitivity
- Stabilisation

Znn.com: Rainbow customizations

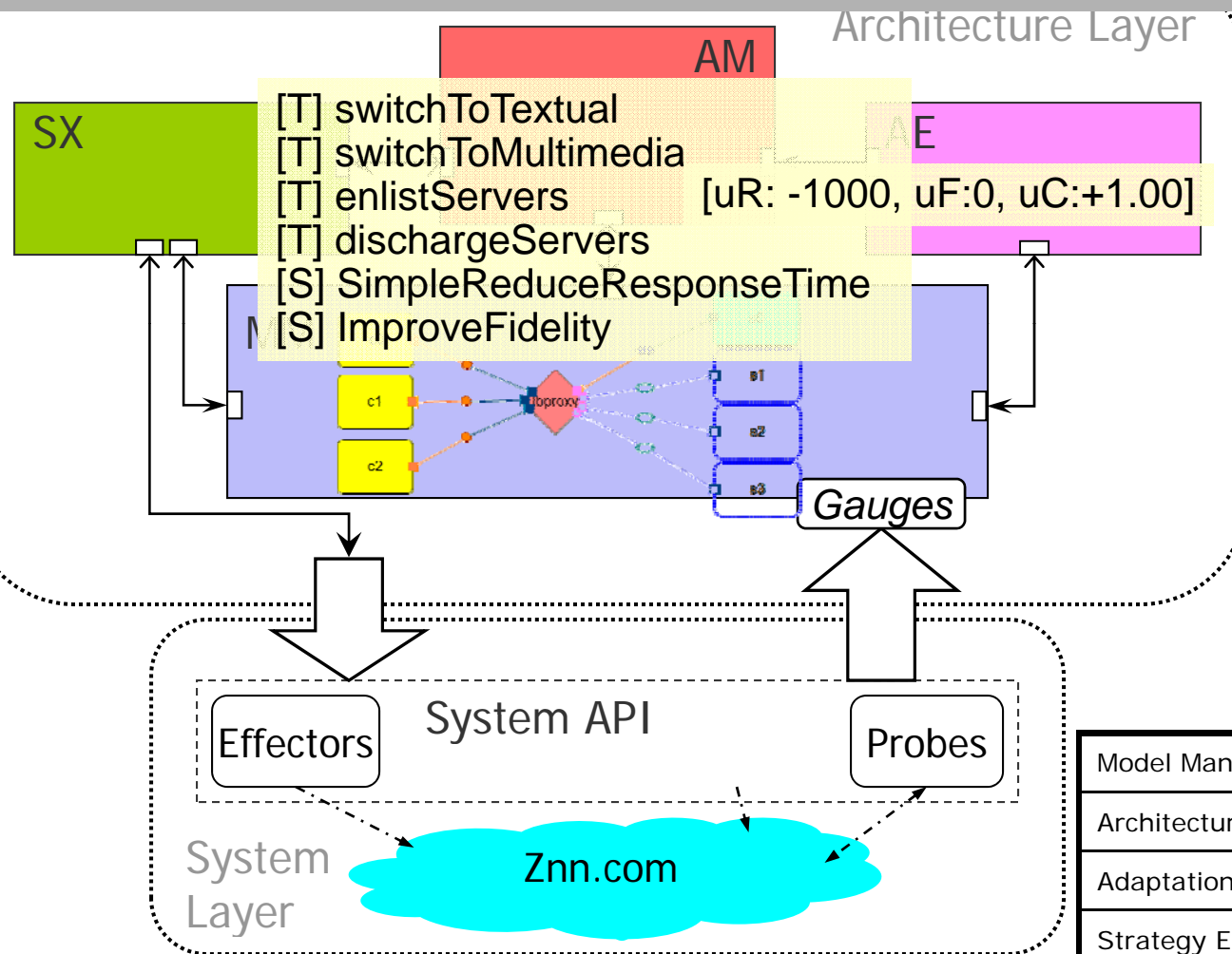


Znn.com: Rainbow customizations

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

uR:
...
utility:
0: 1.00
500: 0.90
1500: 0.50
4000: 0.00

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



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

Data on Customization Effort

No	TalkShoe Customization Task	Duration
1	Target system monitors and effectors	12.9
2	Model capture	4.4
3	Stitch script	8.5
4	Roundtrip integration + modification (small)	8.2
	Total customization clock time:	~34 h
	In man-hours:	~56 h

No	Znn.com Customization Task	Duration
1	Target system monitors and effectors	56.1
2	Model capture	13.3
3	Stitch script	21.3
4	Roundtrip integration + modification (small)	2.7
	Total customization time:	~93 h

Evidence to Fulfill Thesis Claims

- We can provide software engineers the ability to add and evolve self-adaptation capabilities
 - cost-effectively
 - for a wide range of existing software systems and
 - for multiple objectives
- by defining a framework that
 - factors out common adaptation mechanisms and
 - provides explicit customization points to
 - tailor self-adaptation capabilities for
 - particular classes of systems and
 - quality objectives.

General: broad spectrum of systems, typical quality dimensions of concern

Cost-effective: reduced engineering cost relative to existing, specialized solutions

Transparent: adaptation process understandable, actions composable, choice automatable

- **General** – client-server, service-coalition, N-tier
performance, cost, content fidelity, availability, security
- **Cost-effective** – 1st & 2nd-order reuse; 3-6x(dev) / 4-32x(upd) effort savings
- **Transparent** – Znn.com: shows 3 aspects; TalkShoe: shows understandability; sys-admins: validate Stitch concepts and expressiveness

Summary of Evidence

- **Generality**
- **Cost-Effectiveness**
 - Reuse
 - CS → Libra
 - Znn.com → TalkShoe
 - Ease-of-Use
 - **Znn.com (93 hrs)**
 - **TalkShoe (56 hrs** with chief engineer)
- **Transparency**
 - Znn.com & TalkShoe
 - **Sys-admin interviews (Stitch concepts)**
 - **CMU netbwe subsystem (Stitch expressiveness)**

System Styles	Client-Server	Service-coalition	N-Tier
	Quality Dimensions		
Performance	C-S Sys	Libra Video-Conference	Znn.com
Provision cost			
Content quality			
Disruption			
Availability			TalkShoe (commercial)
Security	<i>UnivGradeSys (feasibility)</i>		