Improving Resilience through Analysis and Synthesis of Adaptation Strategies

David Garlan

in collaboration with Javier Cámara, Gabriel Moreno, Bradley Schmerl

Institute for Software Research, Carnegie Mellon University

SoS Lablet Meeting, July 2016

The Problem

- Resilience is an important requirement for modern software-based systems

Maintain high-availability and optimal performance even in the presence of

- system faults
- changes in environment
- attacks
- changes in user needs and context

- Current resiliency mechanisms in systems are error prone, hard to maintain, difficult to verify

How is resilience addressed today?

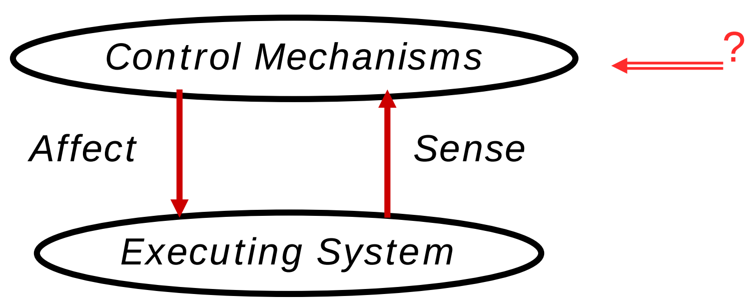
- Low-level, embedded mechanisms
  - Effective and timely
  - Local view of problem state makes it hard to diagnose and correct
  - Brittle to changes in needs/usage
  - Costly to modify after deployment
- High-level, human management
  - Global perspective on problem state
  - Flexible w/res/to changes in policy
  - Costly
  - Error-prone and slow

The Adaptive Systems Approach

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

But how?

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



Model-based Adaptation

Example: Google File System

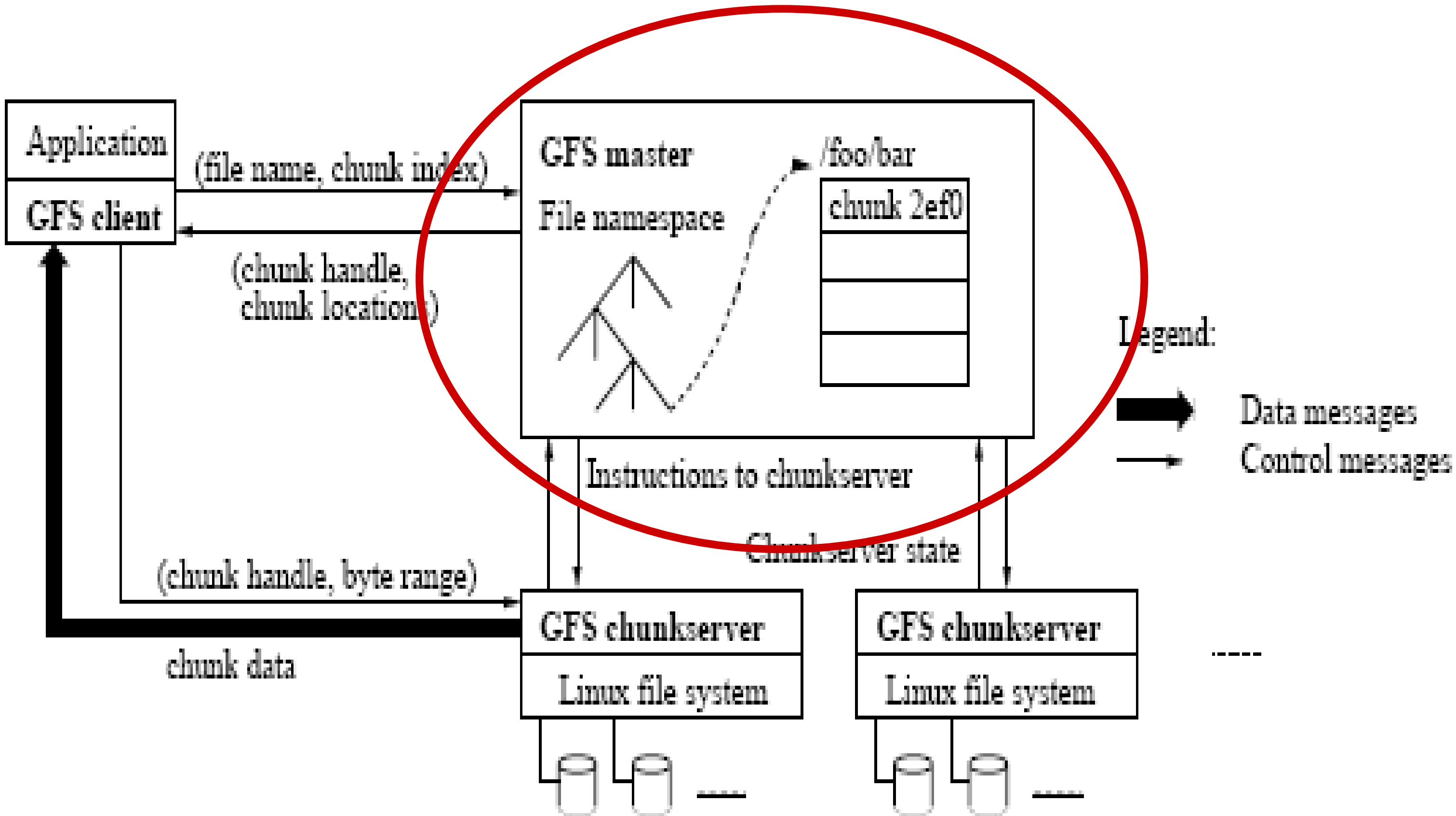


Figure 1: GFS Architecture

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

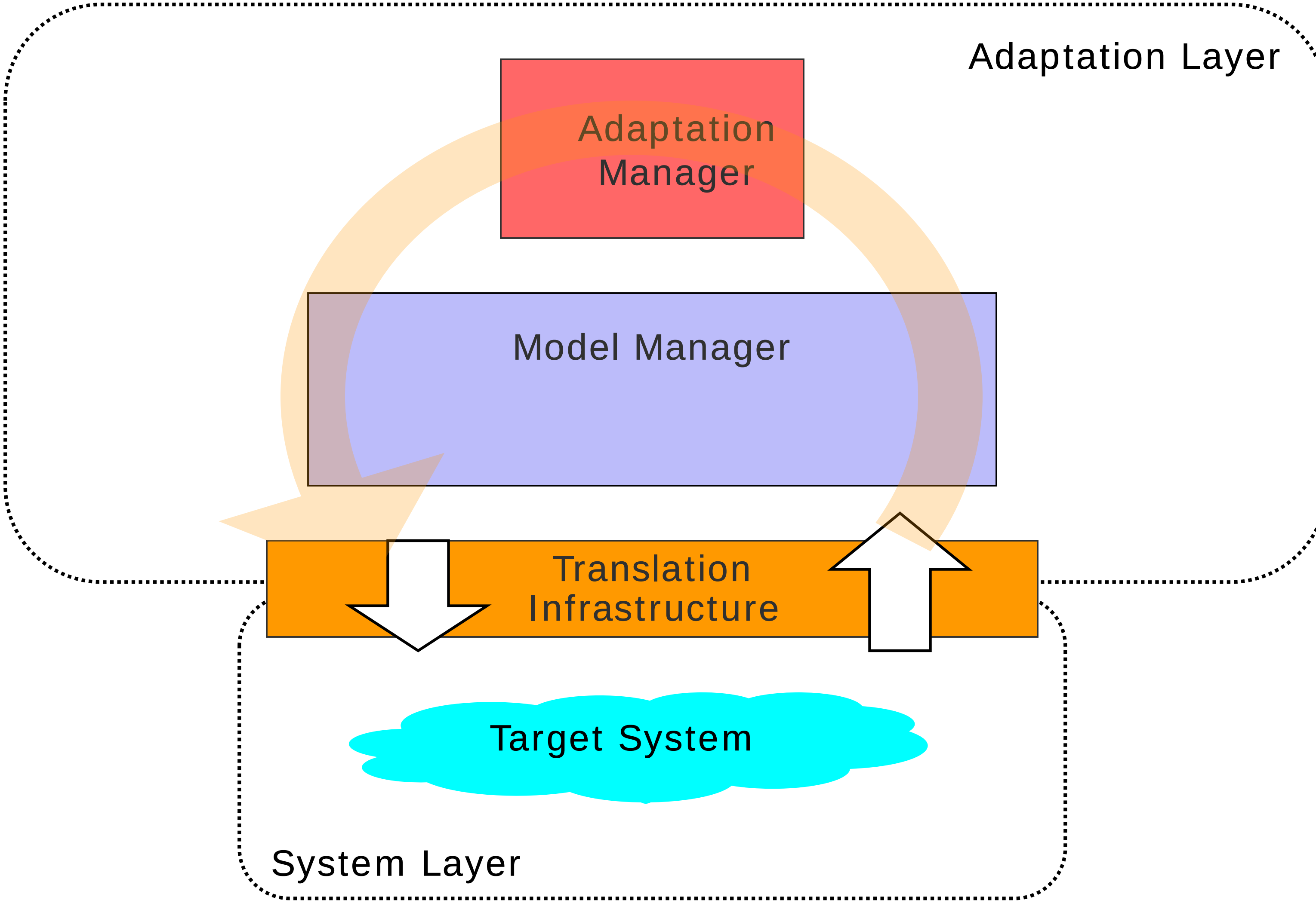
The Self-Adaptation Challenge

- Engineer self-adaptation to support
  - Cost-effectiveness
  - Legacy systems
  - Domain-specific adaptations
  - Multiple quality dimensions
  - Ease of changing adaptation policies
  - Assurance about the effects of self-adaptation actions and strategies

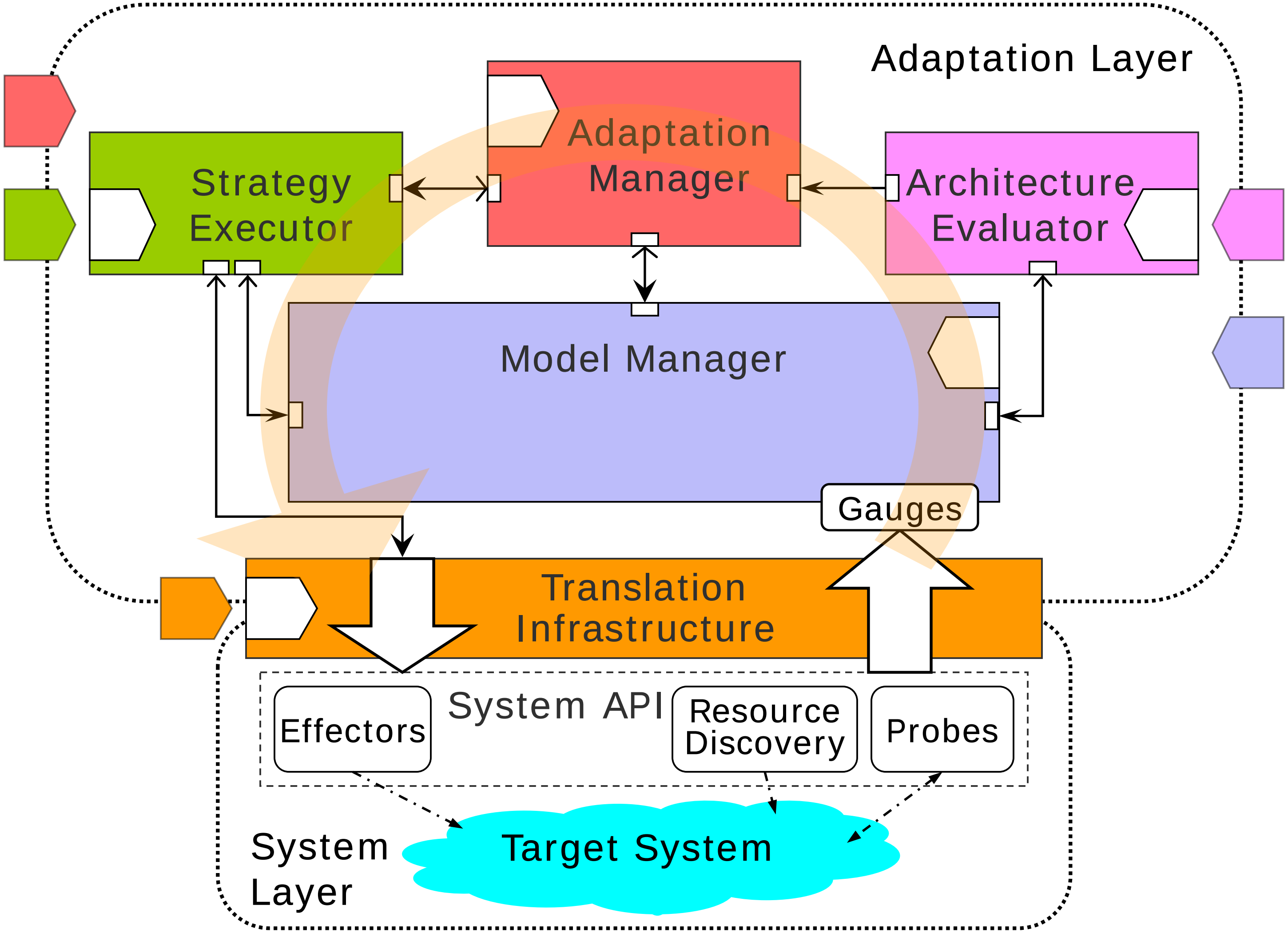
Rainbow

- A framework that
  - Allows one to add a control layer to existing systems
  - Uses dynamically updated architecture models to detect problems and reason about repair
  - Can be tailored to specific domains
  - Separates concerns through multiple extension points: sensors, actuators, models, conditions for adaptation, repair policies
- A language (Stitch) for specifying and reasoning about repair strategies

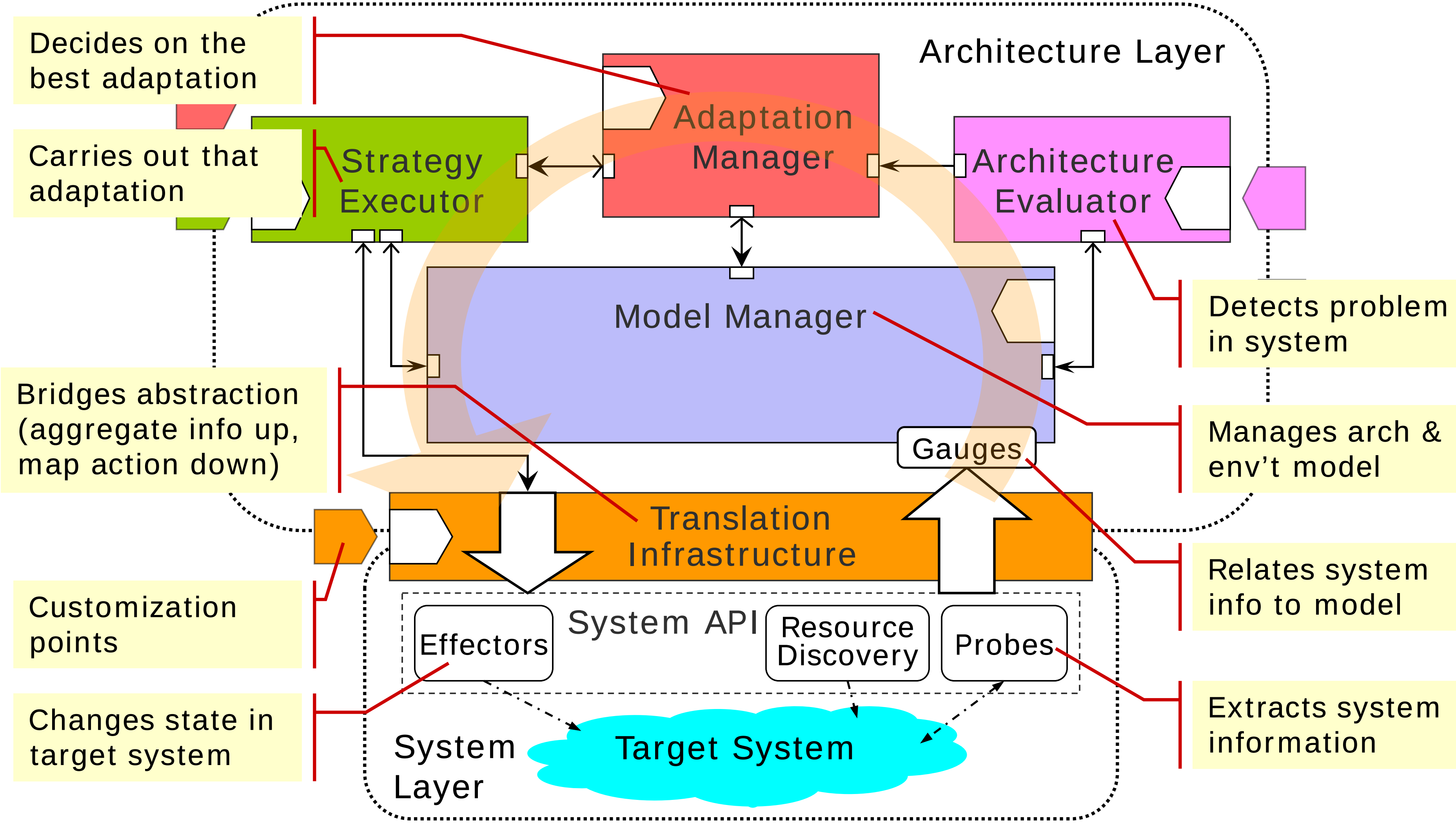
The Rainbow Framework



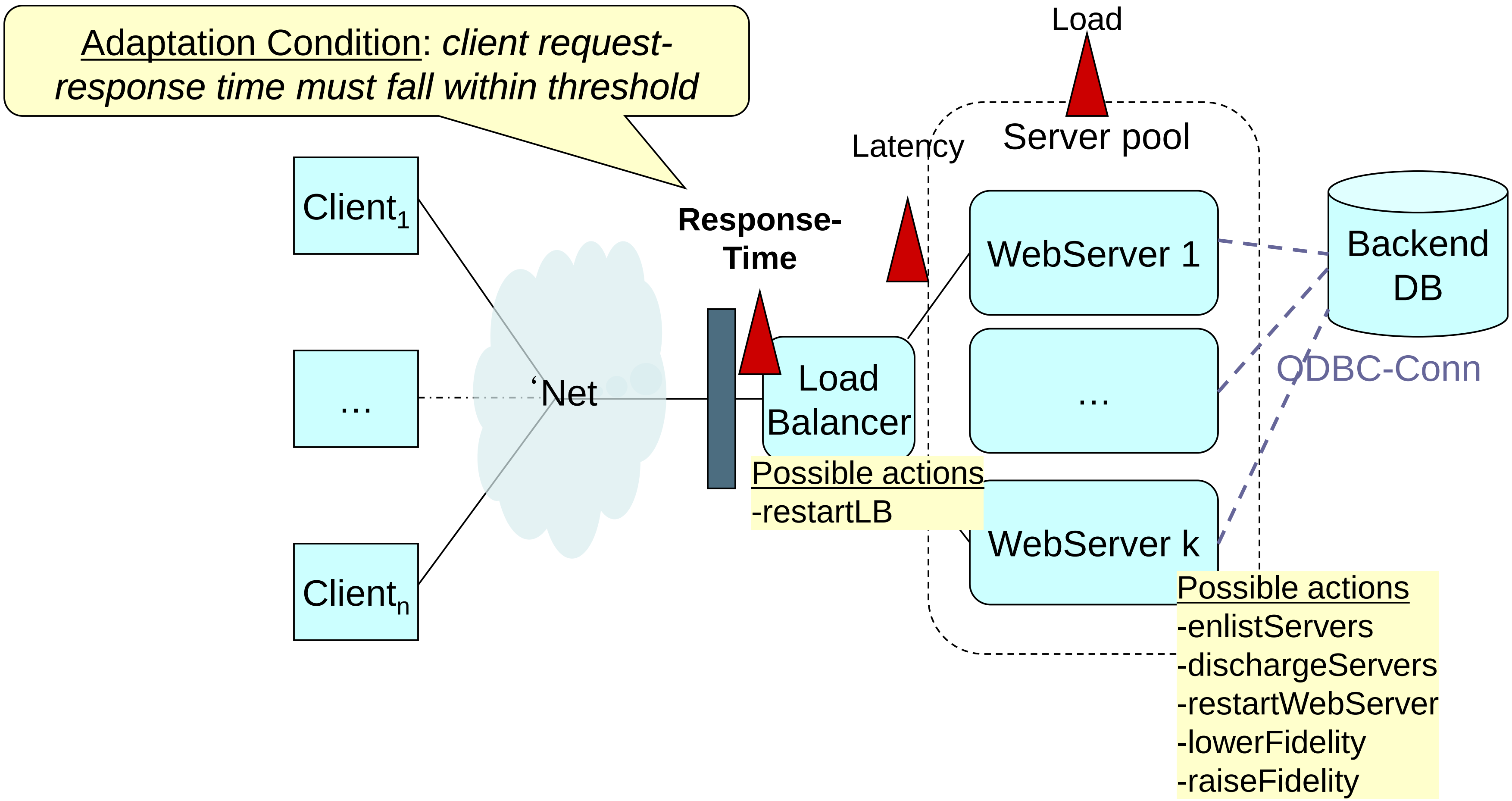
Rainbow Framework Overview



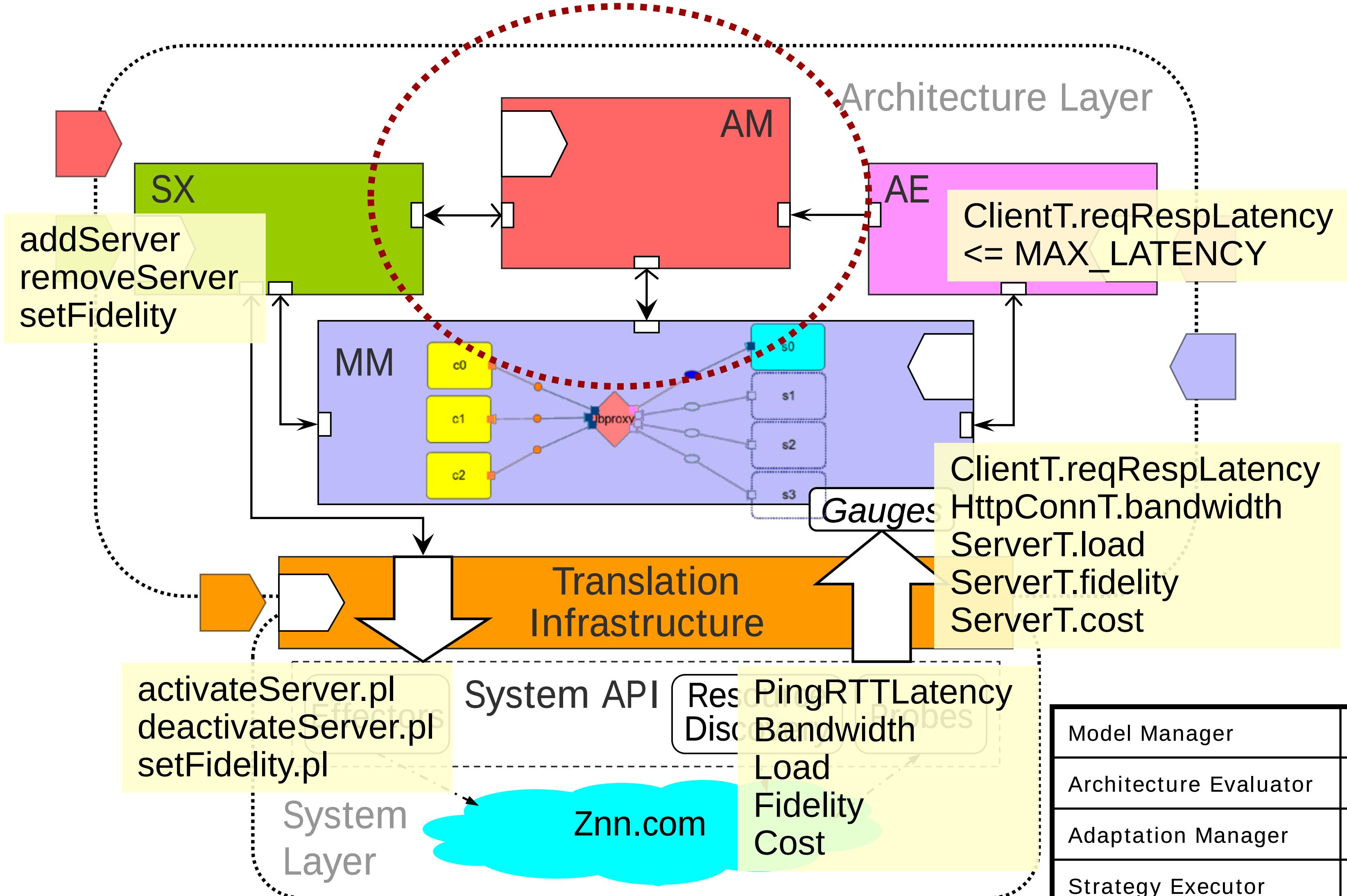
Rainbow Framework Overview



Self-Adaptation Example: *Znn.com*



Znn.com: Rainbow Customizations (Part 1)



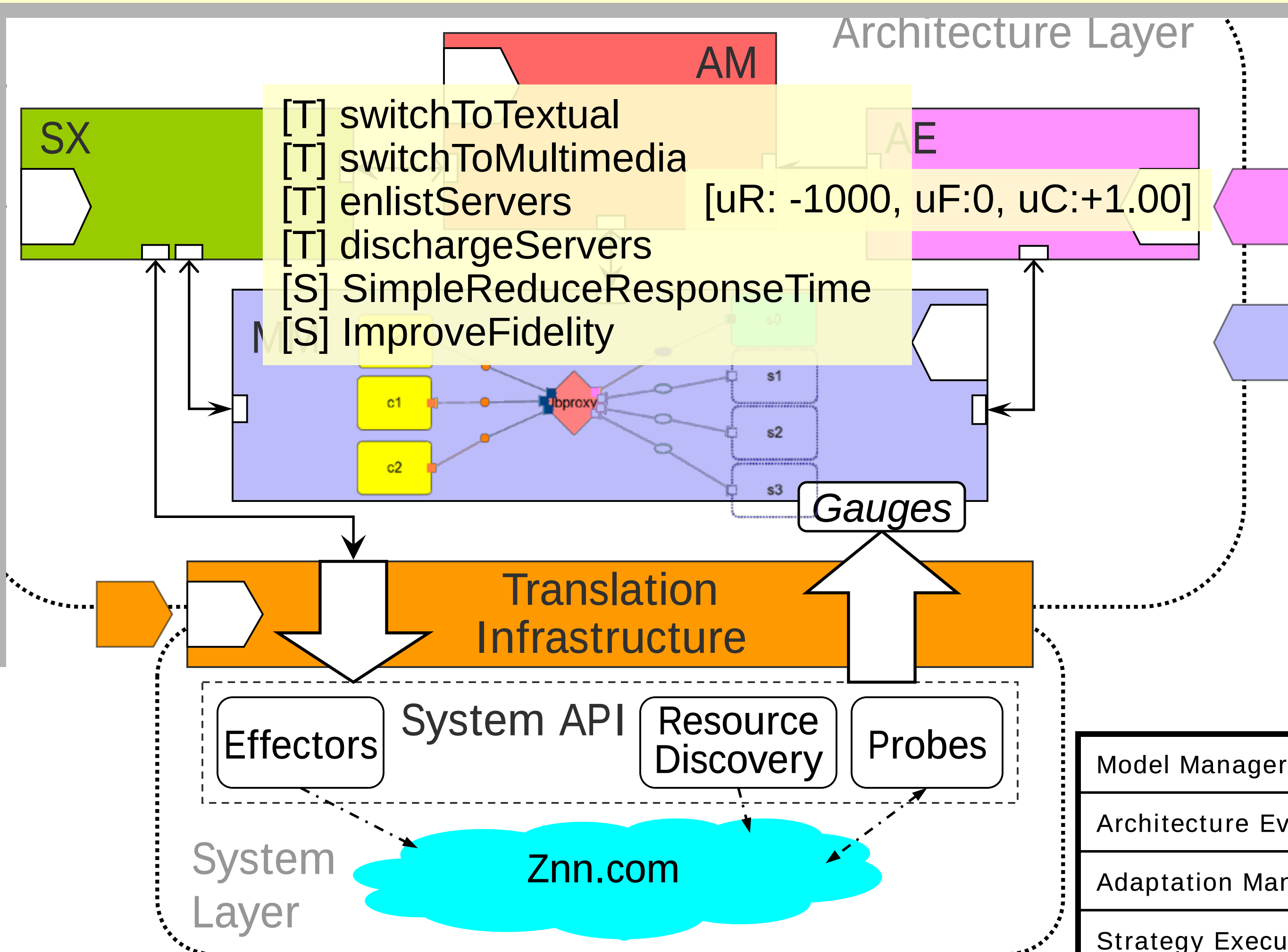
Model Manager	MM
Architecture Evaluator	AE
Adaptation Manager	AM
Strategy Executor	SX

Znn.com: Rainbow Customizations (Part 2)

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

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

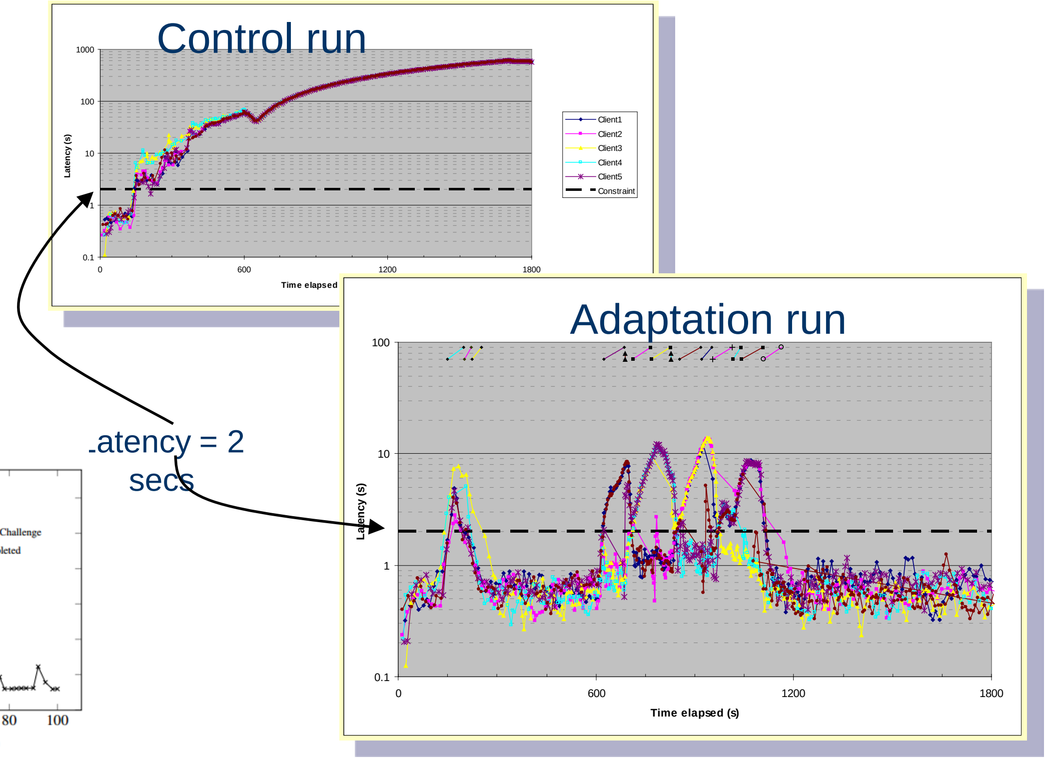
weights:
  uR: 0.3
  uF: 0.4
  uC: 0.2
  uSF: 0.1

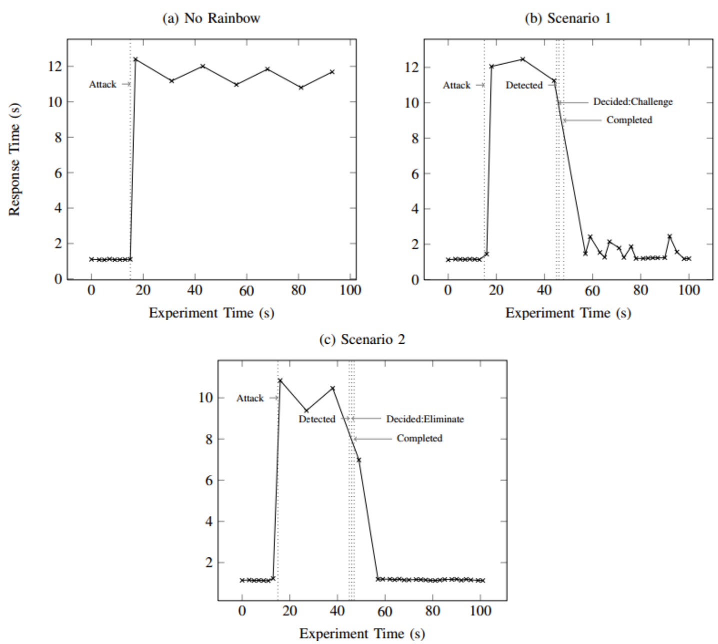


Model Manager	MM
Architecture Evaluator	AE
Adaptation Manager	AM
Strategy Executor	SX

Prior Results

- Manage adaptation balancing cost, performance
- Respond to DoS attacks



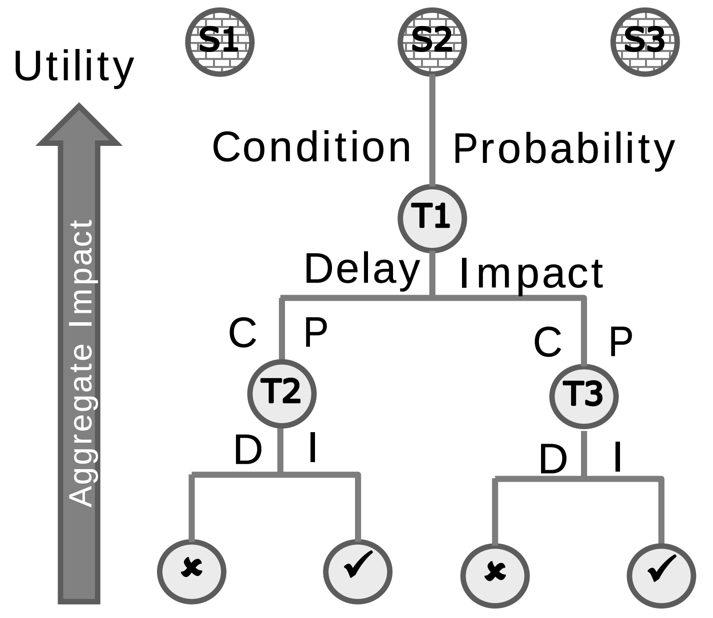


Rainbow Adaptation Decision Overview

- Selection from a set of adaptation strategies
  - Multiple strategies may be applicable at a given time
- Language for expressing strategies as a tree
  - Conditions: when are branches applicable
  - Actions: tactics that will modify the system
- Tree is annotated with properties that permit selection of strategy with highest utility
  - Delays: expected time for effects to be observed
  - Impacts: tactic impacts specified as expected effect of actions on quality dimensions
  - Utility: preferences over the quality dimensions determine utility of impact
  - Uncertainty: Nodes represent probabilistic choices

Stitch: A Language for Specifying Self-Adaptation Strategies

- Control-system model: Selection of next action in a strategy depends on observed effects of previous action
- Uncertainty: Probability of taking branch captures non-determinism in choice of action
- Asynchrony: Explicit timing delays to see impact
- Value system: Utility-based selection of best strategy allows context-sensitive adaptation



Strategy Selection: Example

- 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]**

Algorithm

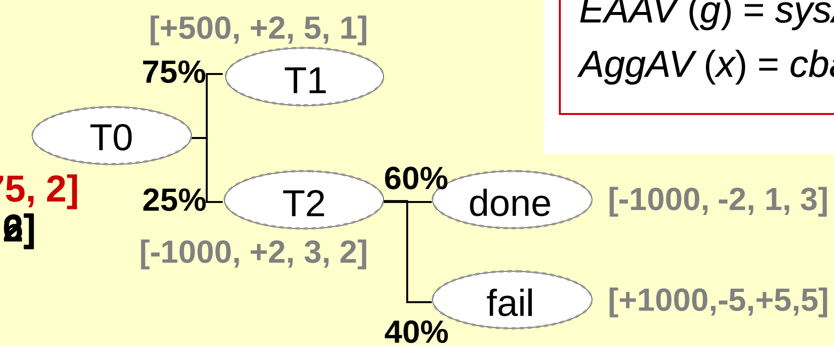
Given tree g with node x and its children c:

$EAAV(g) = sysAV + AggAV(root(g))$

$AggAV(x) = cbav(x) + \Sigma_c\, prob(x,c)\, AggAV(c)$

Score = 0.58

[-125, 1.5, 4.75, 2]



- Propagate cost-benefit vectors up the tree, reduced by branch probabilities
- Merge expected vector with current conditions (assume: [1025, 3.5, 0, 0])
- Evaluate quality attributes against utility functions
- Compute weighted sum to get utility **score**

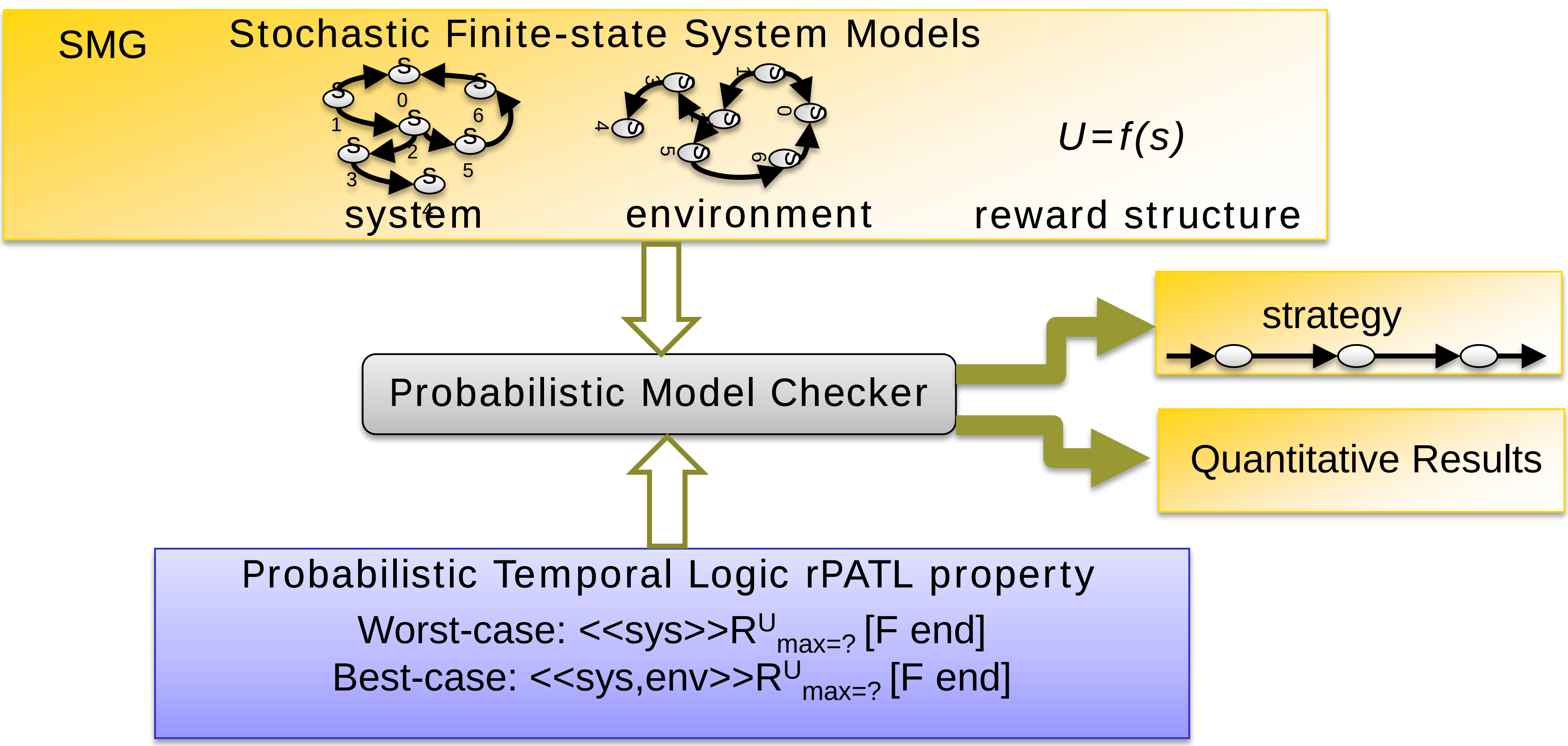
Beyond specification and selection

- Ability to accurately predict outcomes of strategy execution under various environment assumptions (e.g., best-case, worst-case)
- Ability to determine conditions under which different strategies are best suited
- Ability to quantify the value of adding new tactics to the adaptation repertoire
- Ability to synthesize strategies
  - Off-line
  - On-line
- Ability to know when humans should be brought in to the process

Probabilistic Model Checking

- Probabilistic Model Checking: formal verification technique that enables the modeling and (quantitative) analysis of systems that exhibit probabilistic behavior
  - Used in communication protocols, security, biological systems, ...
  - Can capture different sources of uncertainty (e.g., environment changes, outcome of adaptation tactics)
- Stochastic Multiplayer Games (SMGs)
  - Enable modeling and analysis of competitive behavior
  - SMG models include a set of player coalitions that compete against each other to achieve their own goals
  - Each player controls a set of stochastic processes
  - Can be extended with rewards/costs (e.g. time, battery consumption)
  - Can naturally model the interplay between an adaptive system and an adversarial environment
  - Can be used to synthesize optimal strategies

Analysis via Probabilistic Model Checking of SMGs



Strategy Analysis

Goal

Determine the behavioral envelope of the system under adaptation

- Model system and environment as players in a SMG
  - System tries to maximize objective function *f* (e.g., utility, probability of satisfying a property expressed in TL, etc.)
  - Environment can be considered as
    - Adversarial: tries to minimize the value of *f* -> worst-case analysis
    - Cooperative: tries to maximize the value of *f* -> best-case analysis
- Use probabilistic model checking to analyze adaptation performance with respect to *f*
  - Compute the maximum and minimum values of *f* that system player can achieve by following an optimal strategy

Example: Znn.com SMG Model

- Environment player
  - Every period places an arbitrary but bounded amount of request arrivals
- System player
  - Computes average response time using queuing model
  - Selects adaptation tactic non-deterministically
    - Enlist server (latency)
    - Discharge server (no latency)
    - Do nothing

Key idea

No decision algorithm is encoded in the model.

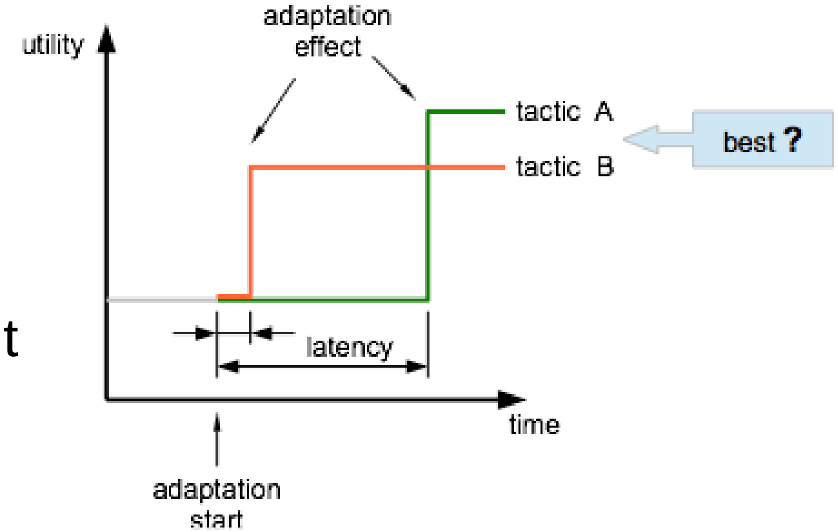
Adaptation strategy based on

• what the system can do (tactics)

• the expected utility of the adaptation

Example: Assessing the Potential Benefit of Proactive, Latency-Aware Adaptation

- Different tactics take different time to produce the intended effect
  - Changing content fidelity: <1s
  - Adding a new Cassandra node: ~3min [Gambi 2013]
- How does the latency of different tactics affect the decision?



Goal

Compare latency-aware (LA) adaptation with non-latency-aware (NLA) adaptation to assess its potential benefit. Synthesize an optimal strategy.

SMG Analysis Results

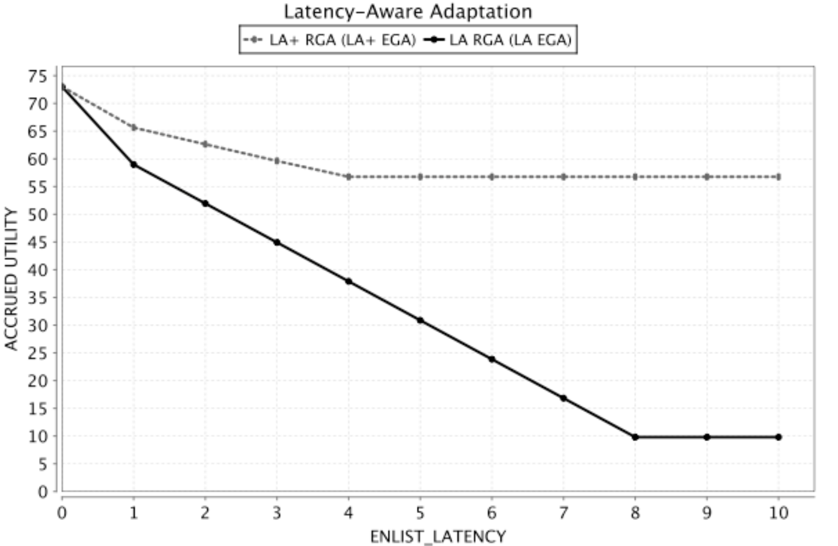
% improvement with latency-aware adaptation in worst- and best-case scenarios

Latency	$\Delta U(\%)$ worst-case	$\Delta U(\%)$ best-case
τ	12.83	21.44
2τ	27.46	22.24
3τ	34.61	23.05

τ : time between adaptation decisions = 10 sec

$\Delta U(\%)$: increase in utility with LA over NLA

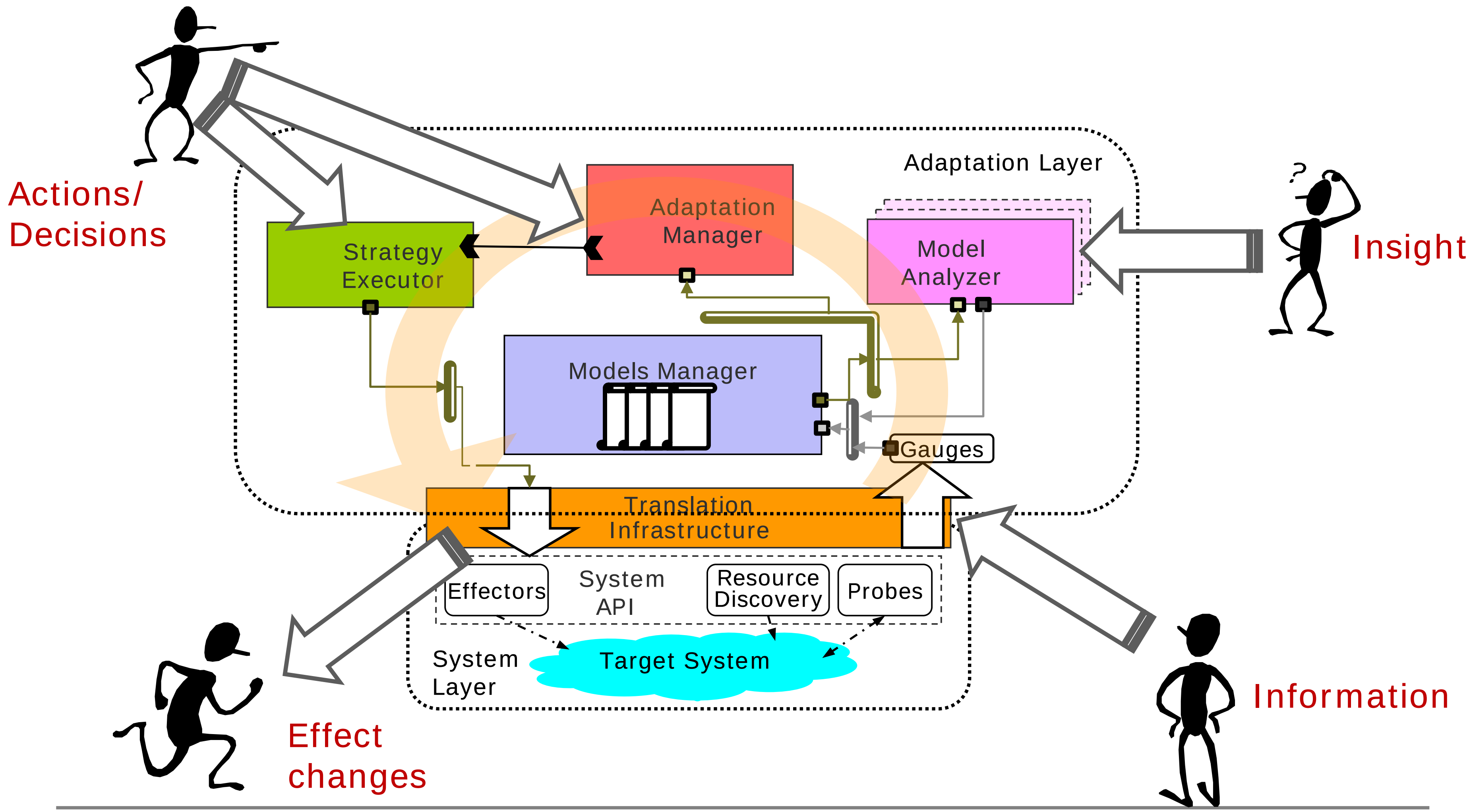
quantify impact of adding new tactics



Reasoning about Human Involvement in Adaptive Systems

- Eliminating human operators is a nice goal, but not always possible
- It is not always possible for a system to automatically adapt for all situations.
- Some actions may require human involvement
- Humans may understand context better than the system

Humans in the Loop

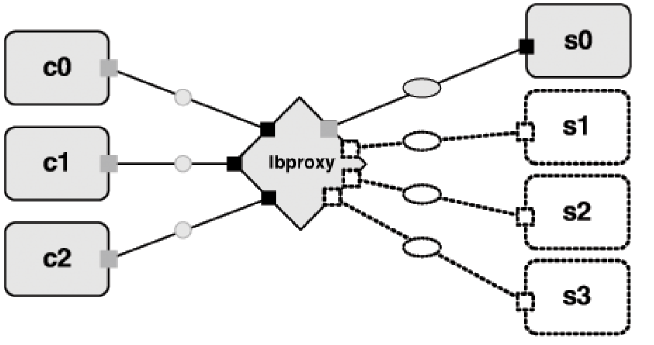


Reasoning about Humans in the Loop

- Human behavior influenced by
  - Changing load and attention
  - Different expertise
  - Physical attributes (access to physical locations, timing)
- Questions that we want to be able to answer
  - What is the likely outcome if a human is involved?
  - Should human participate in the adaptation?
- Framework to formally reason about human involvement in adaptation
  - Focus on **humans as actuators**
  - Extension of language to express adaptation models (Rainbow/Stitch) with human factors (OWC model)
  - Formalization to analyze human-system-environment interactions (Stochastic Multiplayer Games)

Example: Denial of Service in Znn.com

- **Typical news website infrastructure**
  - Pool of replicated servers connected to load balancer
    - Size can be dynamically adjusted
  - Servers can deliver contents with different fidelity levels (text, images, videos...)
    - Content fidelity can be dynamically changed



- **Application layer DoS** (e.g., Slowloris)
- **Quality objectives**
  - Performance: request-response time for legitimate clients
  - Cost: number of active servers
  - Maliciousness: percentage of malicious clients
  - Annoyance: disruptive side effects of tactics

Tactics and Strategies

- **DoS mitigation tactics/strategies selected to provide interesting analytical situations**
  - For example, Adding capacity is much less aggressive than Blackholing, but it is more costly

Tactic	Description
Add capacity	Activate additional servers to distribute the workload
Blackhole ☹	Blacklists clients, requests are dropped
Reduce service	Reduce content fidelity level (e.g., text vs. images)
Throttle ☹	Limits the rate of requests accepted

Strategy	Description
Outgun/Absorb	Combines Add capacity and Reduce service
Eliminate ☹	Combines Blackholing and Throttling

Strategies with Humans: Approach

- Tactics can be automated or manual
- Human actions as tactics
  - Will ask the human to do an action
  - Timing delay gives humans time to do operation
- Model of human sufficient for making the decision
  - Represent characteristics of humans that could affect the decisions
  - Understand how would those characteristics affect the decision about when to involve the human

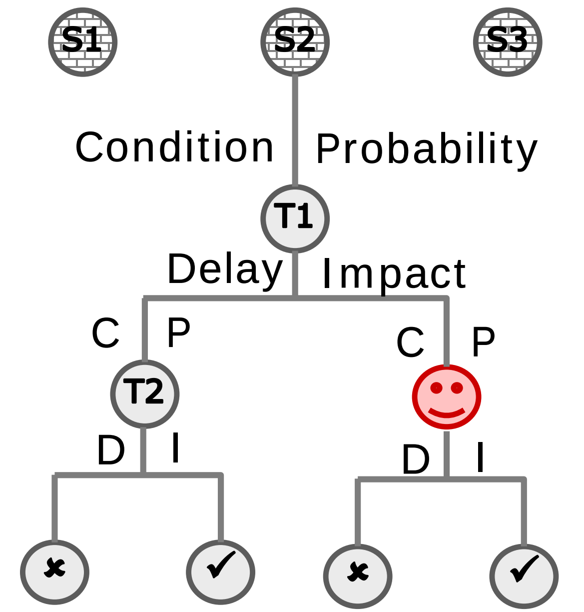
Candidate Model for Human Involvement

- Opportunity-Willingness-Capability model (OWC)*
  - Inspiration from human-cyber systems
- Opportunity:
  - Conditions of applicability for tactics to be done
  - E.g., is human physically located on site? Do they have access to room?
- Capability:
  - How likely the human is to succeed at the task
    - E.g., level of training, seniority, etc.
- Willingness:
  - How likely the human is to do the task
    - E.g., level of attention, stress

*Eskins, Sanders: The Multiple-Asymmetric-Utility System Model: A Framework for Modeling Cyber-Human Systems.

Integrating OWC in Stitch

- Some tactics enact humans
- Opportunity is captured in conditions
- Willingness and Capability affect probabilities
- (Human tactics will likely have longer delays)



OWC Model for blackHoleAttacker (bha)-1

- Opportunity
  - Elements OE={L,B}, where L represents the operator's location:
    - *L.state ∈ {on location (ONL), off location (OFFL)}*

  and B represents whether the operator is busy:
    - *B.state ∈ {busy (OB), not busy (ONB)}*
  - Function: $f_o^{bha}=(L.state==ONL)\ (B.state==ONB)$

Opportunity Elements

```
define boolean ONLNB=exists o:operatorT in M.participants | o.onLocation && !o.busy;
define boolean cHiRespTime=exists c:ClientT in M.components |
c.experRespTime>M.MAX_RESPTIME;

tactic blackHoleAttacker(){
  condition {ONLNB && cHiRespTime;}
  action {ao=Set.RandomSubSet({select o:operatorT in M.participants | o.onLocation && !o.busy},1);
            notify(op, "Blackhole potentially malicious clients");}
  effect {!cHiRespTime;}
}
```

Opportunity Function

OWC Model for blackHoleAttacker (bha)-2

- Willingness
  - Elements WE={S}, where S represents the operator's stress level:
  - Function: $f_w^{bha}=pr_w(S.state)$, *with* pr_w *->* *[0,1]* maps stress levels to probability of the tactic being carried out
- Capability
  - Elements CE={T}, where T represents the operator's level of training.
  - Function: $f_c^{bha}=pr_c(T.state)$, *with* pr_c *->* *[0,1]* where pr_c maps training levels to probabilities of successful tactic performance.

institute for SOFTWARE RESEARCH

Example: Strategies to Absorb/Eliminate excess traffic

Fully automated

```
strategy Outgun
[cHiRespTime] {
   t0 : (cHiRespTime) -> enlistServers(1)@[30000 /*ms*/] { // enlist server, wait 30s
      t1: (sucess) -> done;
      t2: (fail) -> lowerFidelity() @[2000 /*ms*/] {
            t2a: (success) -> done;
            t2b: (fail) -> TNULL;
      }
   }
}
```

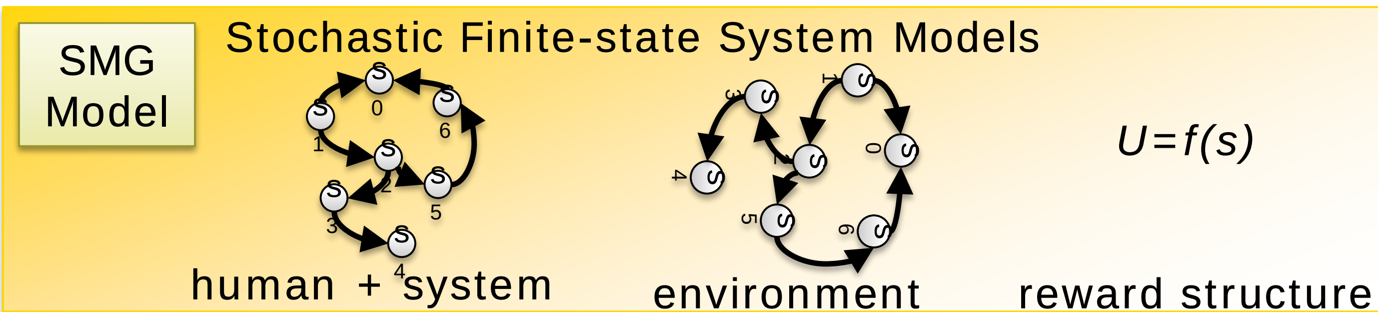
Relies on human effectors

```
strategy Eliminate
[ONLNB & (unhandledMalicious || unhandledSuspicious)] {
   t0: (unHandledMalicious) -> : blackHoleAttacker()@[300000 /*ms*/] { // blackhole, wait 5 min
      t0a: (success) -> done;
      t0ab: (unhandledSuspicious) -> throttleSuspicious()@[30000 /*ms*/] {
            t1a: (success) -> done;
             t1b: (fail) -> TNULL;
             }
      }
}
```

Under what conditions will one strategy be better than the other?

Probabilistic Modeling for HuIL

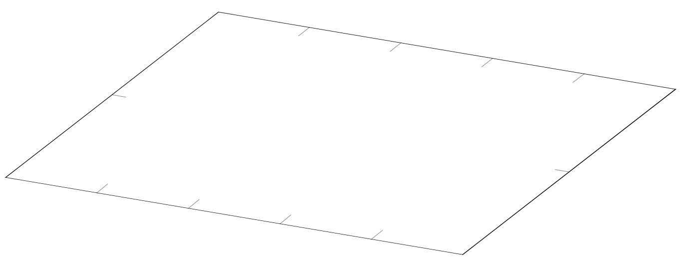
- Model system-human-environment interactions as a Stochastic Multiplayer Game*
  - System + human player tries to maximize utility
  - Environment player tries to minimize utility
    - Enables worst-case scenario analysis



- Quantification of maximum utility that system+human player can obtain
- Synthesis of optimal player strategies
  - Insight about best combined operator/automatic adaptations
  - Context-sensitive notion of optimality

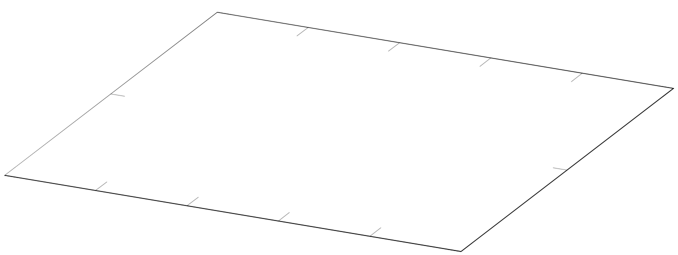
*J. Cámara, G.A. Moreno, D. Garlan: ***Reasoning about Human participation in Self-Adaptive Systems***. 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2015)

Analysis results: varying **C**apability elements (Scenario 1- eliminate malicious clients)



Outgun vs Eliminate accrued utility

Analysis results: varying **C**apability elements (Scenario 2 – optimize user experience)



Outgun vs Eliminate accrued utility

Analysis results: strategy selection (Scenario 1 – eliminate malicious clients)

Eliminate predominates. Human involvement useful even if training is limited, or with low level of malicious clients (20%) if training is good.

institute for SOFTWARE RESEARCH

Analysis results: strategy selection (Scenario 2 – optimize user experience)

Outgun predominates. Human involvement only useful if operator has extensive training (>0.55) and malicious clients >50%.

Current and Future Work

- Adaptation and architecture models can be combined with formal techniques to improve the predictability of SAS in a systematic manner
  - Latency-aware proactive adaptation [SEAMS14]
  - Self-protecting systems
    - Adversarial environments that might include potential attackers
      - Denial-of-Service Mitigations [HotSoS14]
      - Moving Target Defense [MTD14]
  - Human-in-the-loop
    - System formed by a coalition of adaptation manager and human participants [SEAMS15]
- Future work
  - Explicit resolution of uncertainty
  - Machine learning to improve models at run time
  - Combining off-line and on-line analysis/synthesis.

References

- [SEAMS14] Javier Cámara, Gabriel A. Moreno and David Garlan. Stochastic Game Analysis and Latency Awareness for Proactive Self-Adaptation. In *9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, Hyderabad, India, 2-3 June 2014.
- [HotSoS14] Bradley Schmerl, Javier Cámara, Jeffrey Gennari, David Garlan, Paulo Casanova, Gabriel A. Moreno, Thomas J. Glazier and Jeffrey M. Barnes. Architecture-Based Self-Protection: Composing and Reasoning about Denial-of-Service Mitigations. In *HotSoS 2014: 2014 Symposium and Bootcamp on the Science of Security*, Raleigh, NC, USA, 8-9 April 2014.
- [MTD14] Bradley Schmerl, Javier Cámara, Gabriel A. Moreno, David Garlan and Andrew Mellinger. Architecture-Based Self-Adaptation for Moving Target Defense. Technical report, CMU-ISR-14-109, *Institute for Software Research, Carnegie Mellon University*, 2014.
- [FACS14] Javier Cámara, Antonia Lopes, David Garlan and Bradley Schmerl. Impact Models for Architecture-Based Self-Adaptive Systems. In *Proceedings of the 11th International Symposium on Formal Aspects of Component Software (FACS2014)*, Bertinoro, Italy, 10-12 September 2014.
- [SEAMS15] J. Cámara, G.A. Moreno, D. Garlan*:* Reasoning about Human participation in Self-Adaptive Systems. 10th Intnl Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2015)