Model-based Self Adaptation

David Garlan and Bradley Schmerl
Carnegie Mellon University

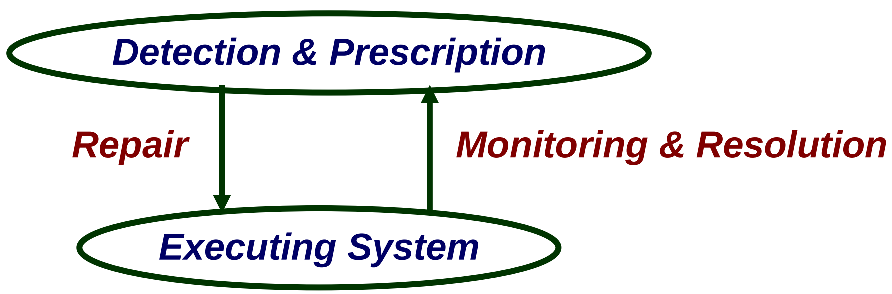
**Workshop on
Self-Healing Systems
(WOSS'02)**

**November 18-19, 2002
Charleston, SC**

Self-Healing Systems?

Systems have built-in mechanisms for

- **Monitoring:** observing run time behavior
- **Resolution:** interpreting the meaning of run time observations
- **Detection:** recognizing when there is a need or opportunity for adaptation
- **Prescription:** identifying a strategy to effect an improvement
- **Repair:** mechanisms to change the system as it is running



Architecture-based Adaptation

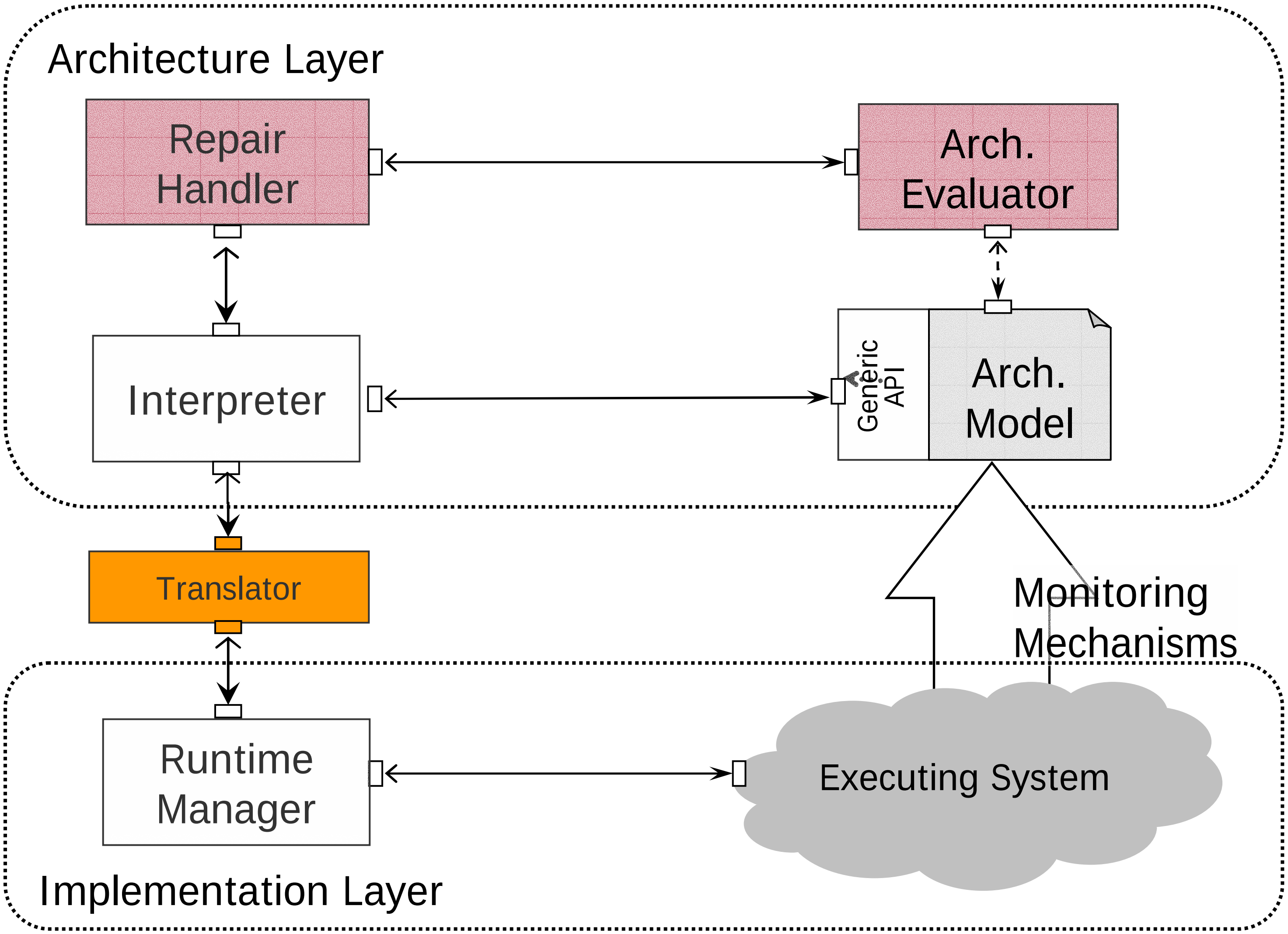
Architectural models are used as basis for resolution, detection & prescription

- Form the basis for control
- Represent the system in terms of gross decomposition into components and connectors
- Allow repairs at the global system level for qualities such as reliability, performance, interoperability

These models are external to the system

- Separate from the system itself
- Require bridging mechanisms
- Provide clean separation of concerns

Architecture-Driven Adaptation



Key Challenge: One size does not fit all

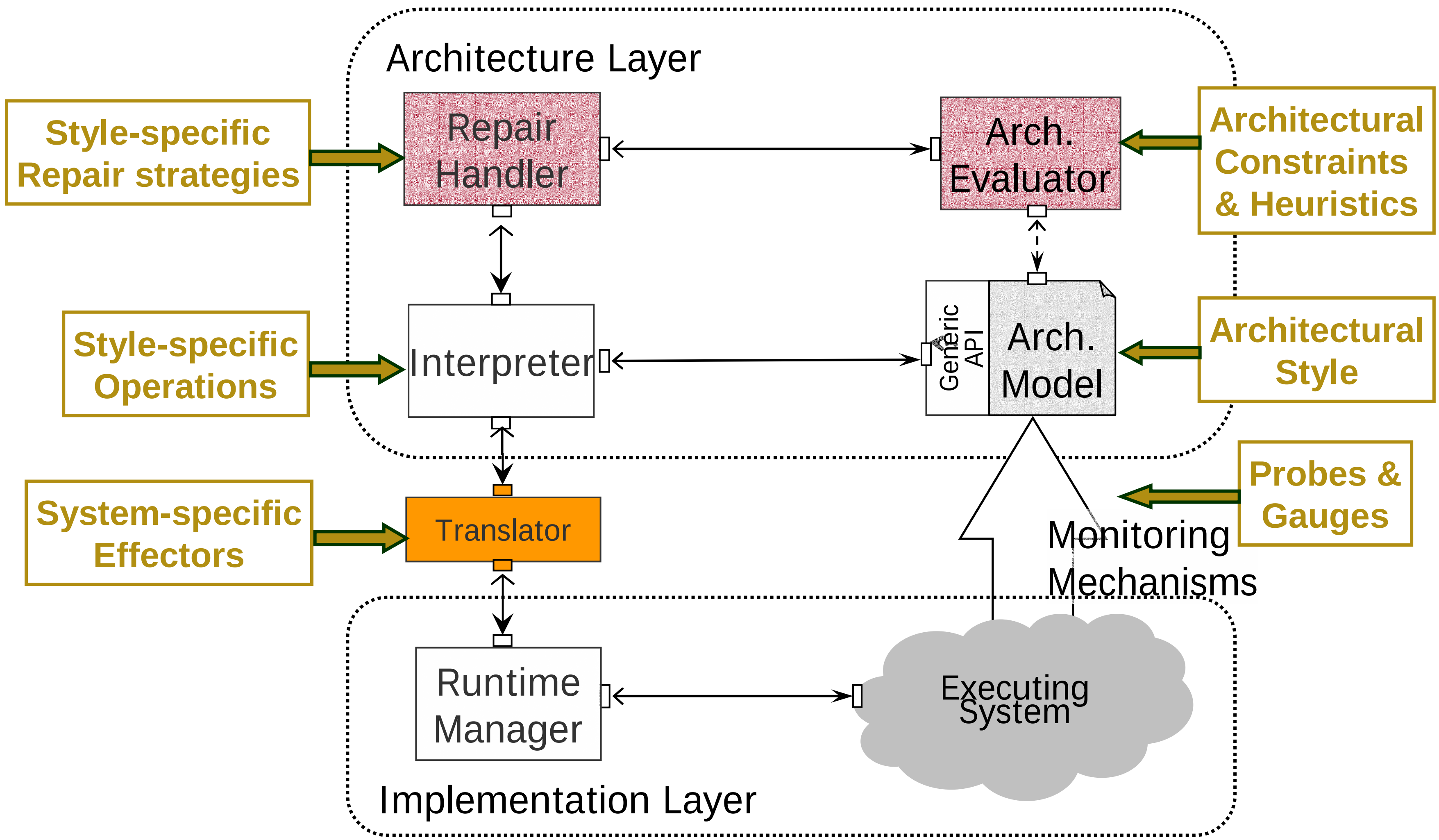
Self-healing must be tailorable to different

- Implementation styles
- Architecture styles
- Qualities of concern
- Repair expertise

Examples:

- A data flow system with end-to-end latency concerns (e.g., video teleconferencing), versus
- A real-time shared variable system with schedulability concerns (e.g., automotive systems), versus
- A blackboard-based planning system with reliability and resource consumption concerns (e.g., a NASA Mars Rover)

Our Approach: Parameterized Framework



Generic Framework

Specific Adaptations

Current:

- Performance adaptation for distributed client-server systems
- Protocol monitoring

Under development:

- Service coalitions for pervasive computing environments
- Shared variable systems (Ford, NASA)

Currently not addressing:

- Probe technologies
- Implementation change mechanisms
- Constraint inference

Some Research Problems

Architectural “recovery” at run time.

Efficient, scalable constraint evaluation

Environment modeling and scoping

Handling multiple models and dimensions of concern

Reasoning about the correctness of a repair strategy

Timing issues

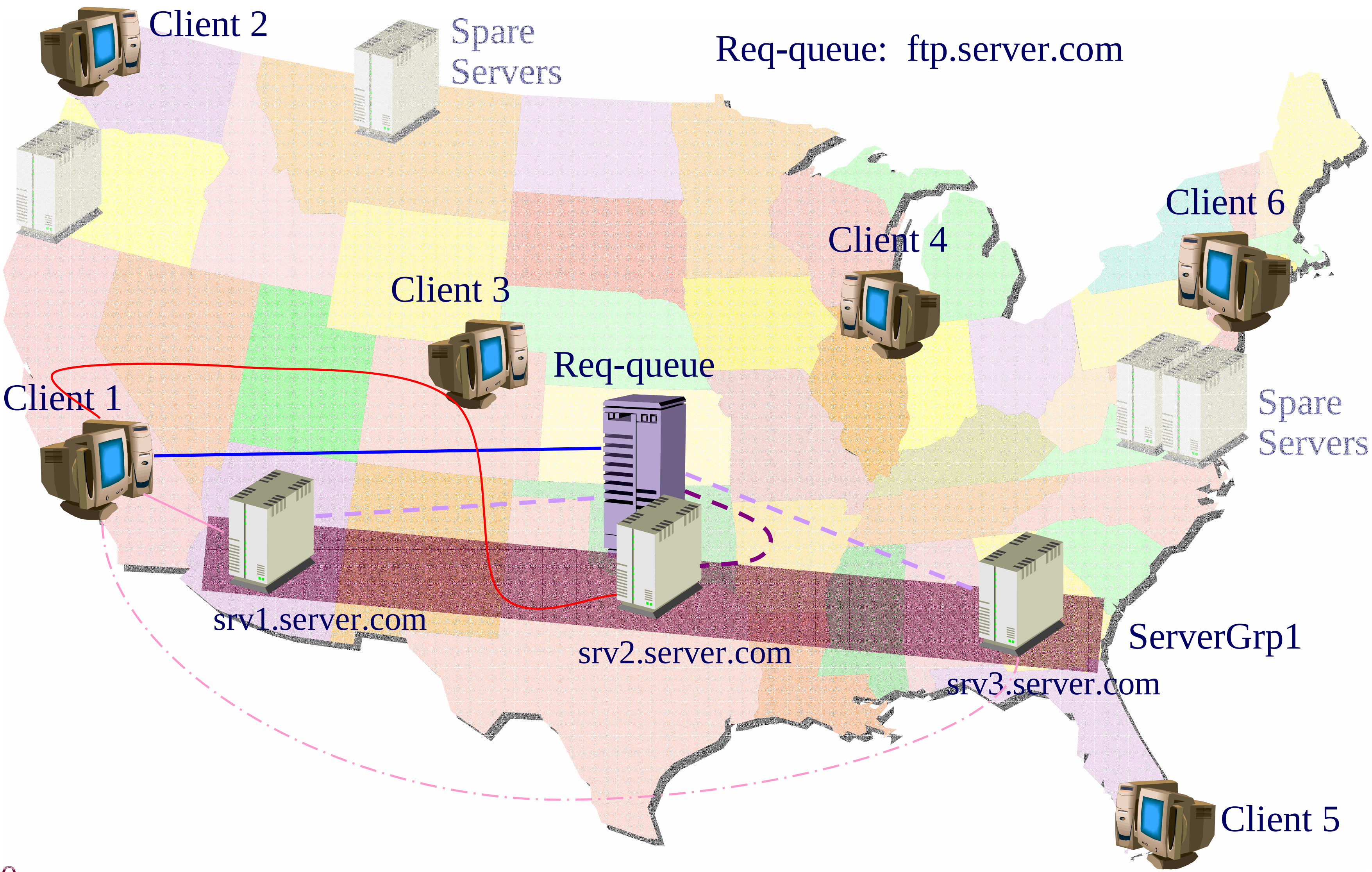
- Non-deterministic arrival of system observations
- Change latencies

Avoiding thrashing

Adapting the adaptation strategies

Last Slide

Example


Client 2
Spare
Servers
Req-queue: ftp.server.com
Client 6
Client 4
Client 3
Req-queue
Client 1
Spare
Servers
srv1.server.com
srv2.server.com
ServerGrp1
srv3.server.com
Client 5

Software Architecture

Graph of interacting components

- Components
- Connectors

Properties capture semantics

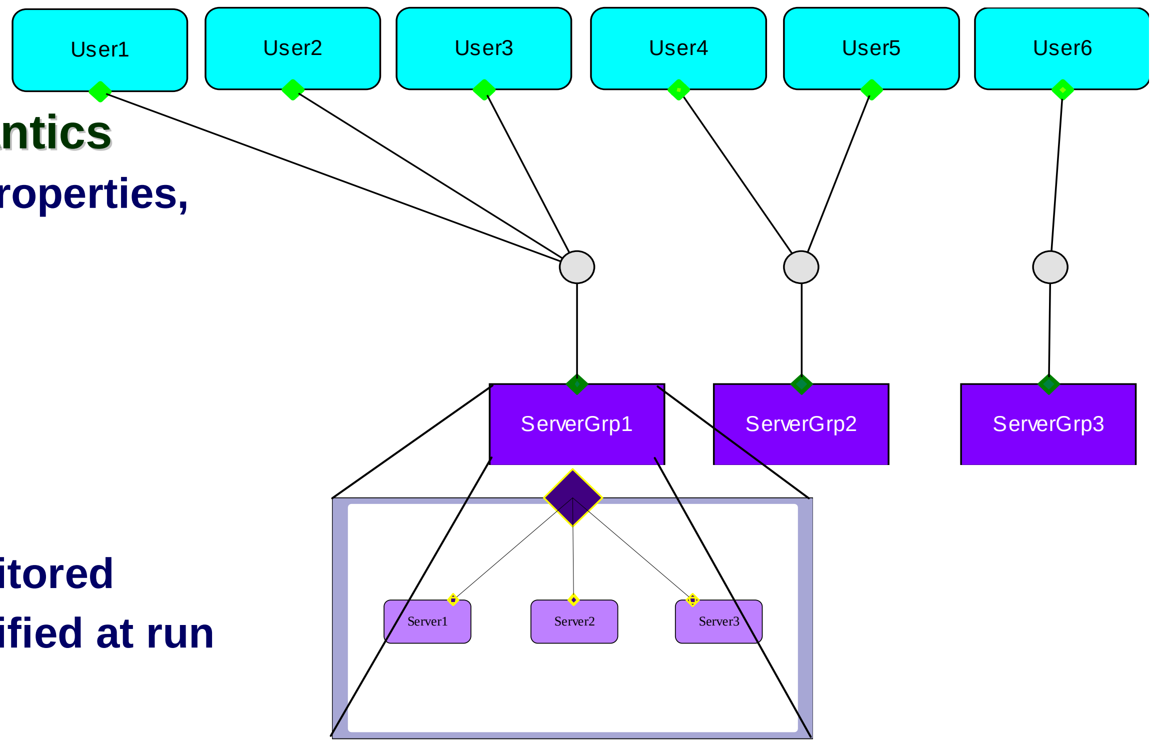
- E.g., performance properties, protocols

Tools to analyze

- Style conformance
- QoS conformance

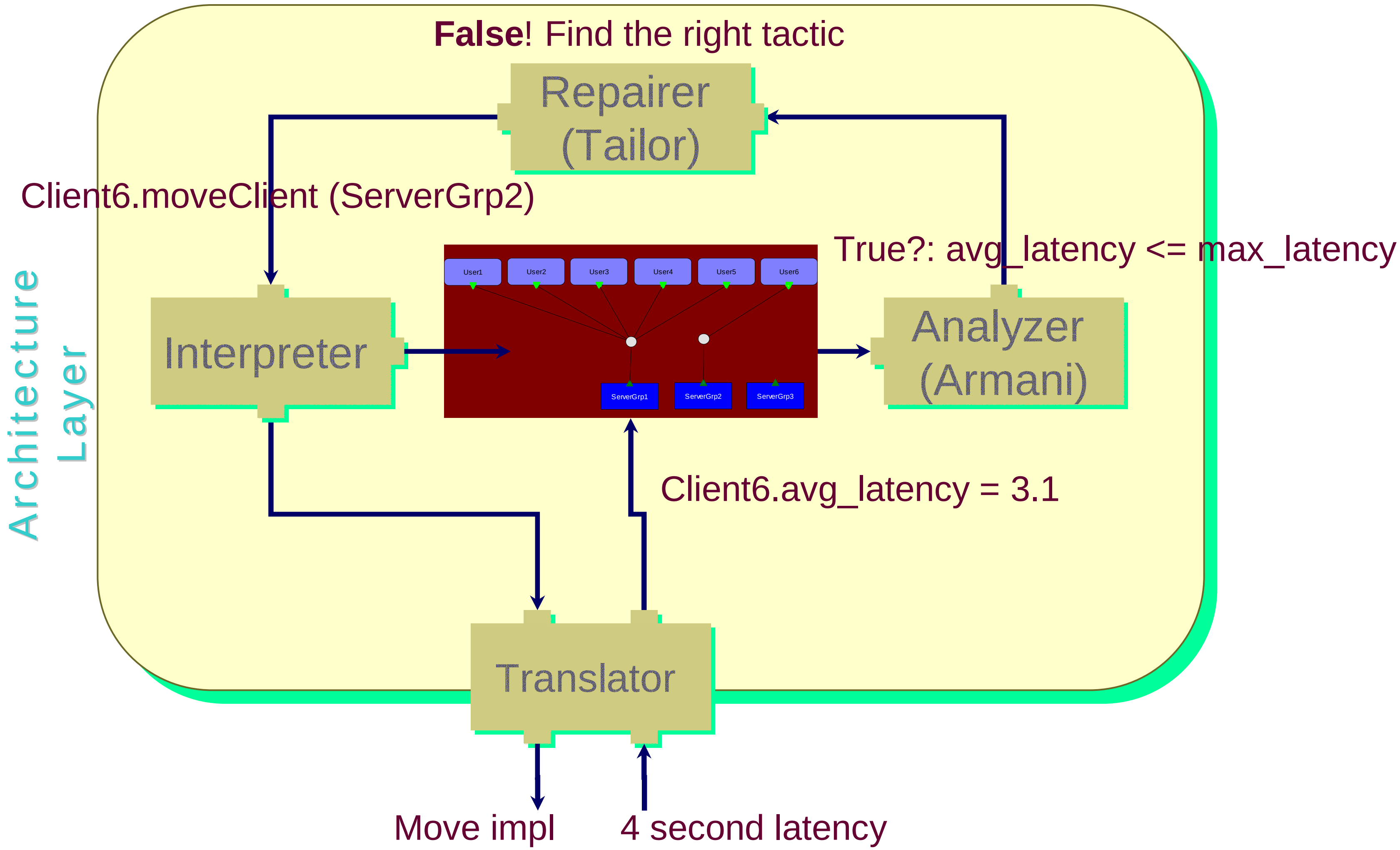
Assumptions

- System can be monitored
- System can be modified at run time

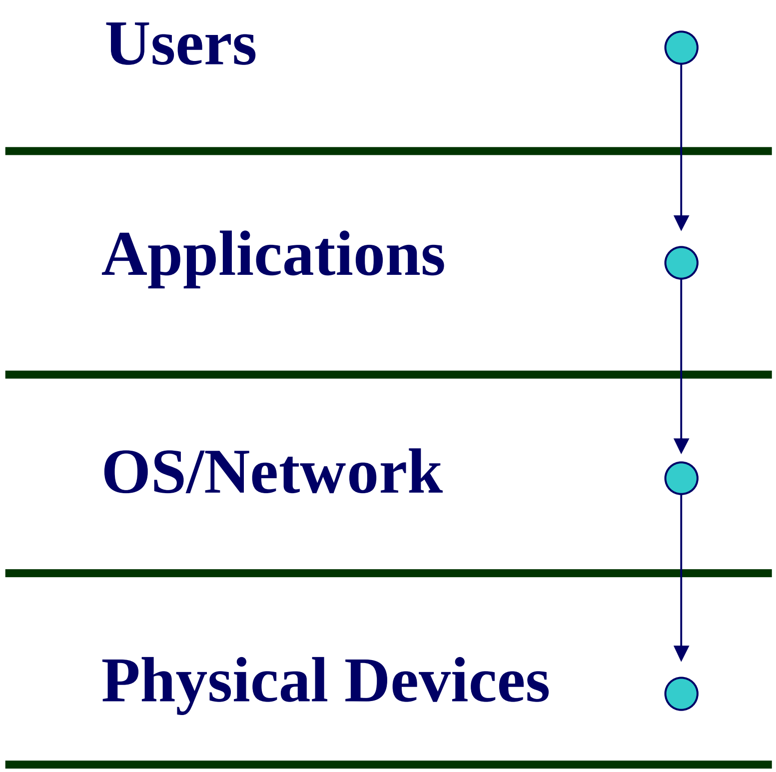


Component ServerGrp1 (ServerGrpRep)

Making Repairs



Today



Tomorrow