

Patterns of Self- Management

Dave Wile

Teknowledge Corp.

Dwile@teknowledge.com

Talk Summary

- What does self-management mean to you?
 - Support for system adaptation to vary the extent to which it satisfies its *designers'* desires based on the dynamic environment
 - Support for system adaptation to vary the extent to which it satisfies its *users'* desires based on the dynamic environment
- What aspects of the self-management problem are you addressing?
- What aspects are you NOT dealing with?
- What domains, properties, or applications are you targeting?
- What are the top two/three new technical ideas/approaches that you are pursuing in this work?

Talk Summary

- What does self-management mean to you?
- What aspects of the self-management problem are you addressing?
 - Externalized view of self-management activities
 - Specification of add-ons needed to
 - Instrument
 - Monitor
 - Decide
 - Effect
 - Cataloging well-known idioms for these activities as patterns
- What aspects are you NOT dealing with?
- What domains, properties, or applications are you targeting?
- What are the top two/three new technical ideas/approaches that you are pursuing in this work?

Talk Summary

- What does self-management mean to you?
- What aspects of the self-management problem are you addressing?
- What aspects are you NOT dealing with [here]?
 - Refinement or implementation of concepts
 - Appropriateness of concepts in different situations
 - Variations of patterns
 - Specific domains where more appropriate idioms would occur
- What domains, properties, or applications are you targeting?
- What are the top two/three new technical ideas/approaches that you are pursuing in this work?

Talk Summary

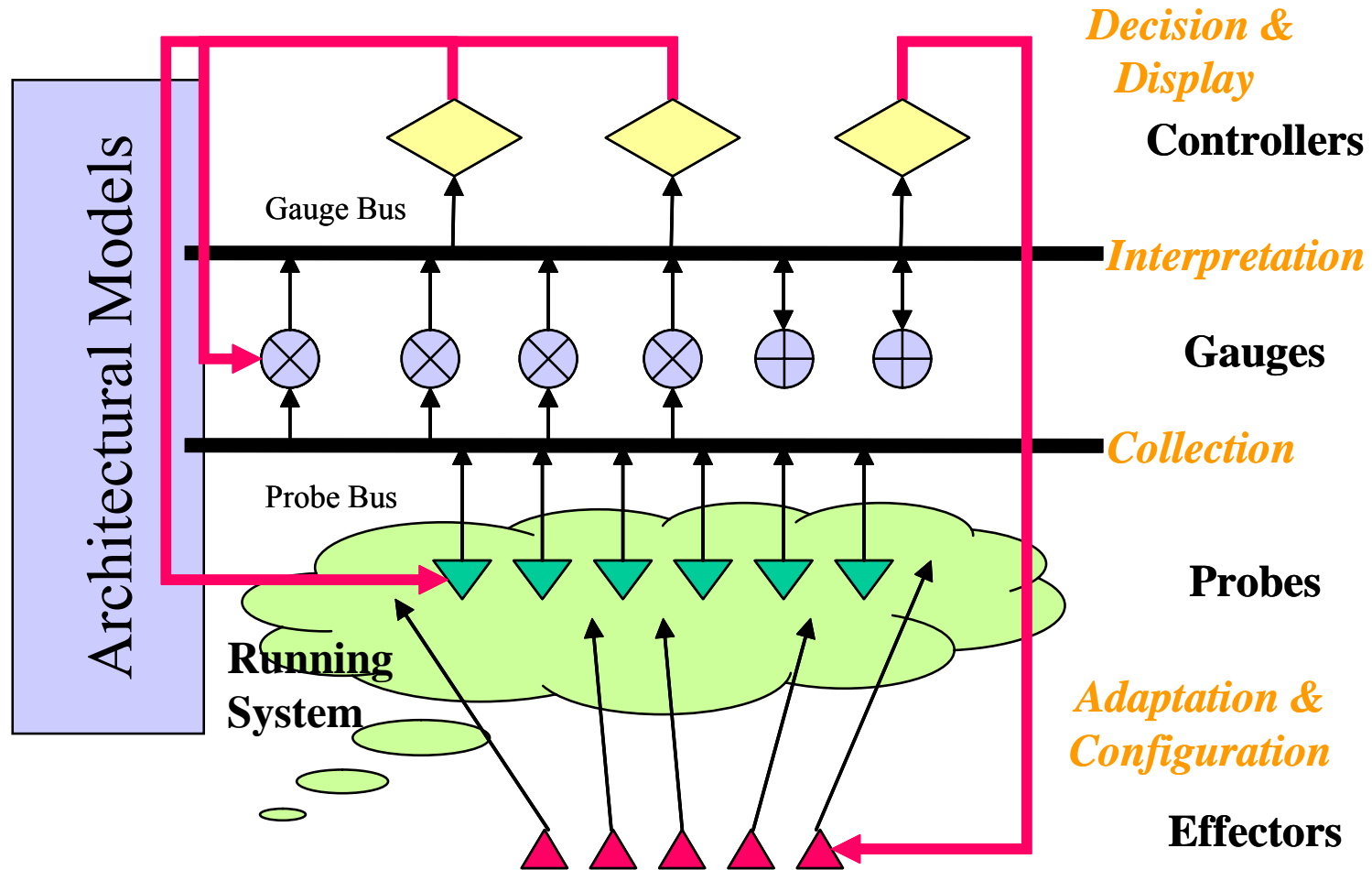
- What does self-management mean to you?
- What aspects of the self-management problem are you addressing?
- What aspects are you NOT dealing with?
- What domains, properties, or applications are you targeting?
 - Coarse-grained systems
 - Not tightly-coupled systems
 - Not highly dynamic / rapidly evolving systems
 - Not closed-off, inaccessible systems (e.g. single monolithic applications)

(I just have not thought about idioms there)
- What are the top two/three new technical ideas/approaches that you are pursuing in this work?

Talk Summary

- What does self-management mean to you?
- What aspects of the self-management problem are you addressing?
- What aspects are you NOT dealing with?
- What domains, properties, or applications are you targeting?
- What are the top two/three new technical ideas/approaches that you are pursuing in this work?
 - Externalized infrastructure
 - Self-Management architectural style
 - Self- Management patterns expressed in the style

Externalized Infrastructure



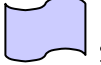
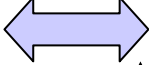














(Source: DASADA II proposal)

Problems

- C2-like: Best for implementation of very dynamic harnesses, where new gauges are created and swapped in and out
- Incapable of expressing direct communication
 - Obfuscates component relationships
 - Obfuscates connection types
- Cannot express implicit coupling relationships
- Difficult to reason about

ALTERNATIVE: Specific architectural style

- Sensors'  information collected by 
- Gauges ,  and 
 - Accumulating  information from other gauges
 - Which are InterpretedBy 
- Interpreters, which are either
 - User Displays 
 - Or Controllers 
 - Which Configure  Gauges, Sensors or Effectors 
 - Or Decide  which Effectors to enable
- Abstractions  are used to model information, written  and read  by all non-system elements, i.e. all but sensors and effectors
- Some sensors' and effectors' activities are coupled 

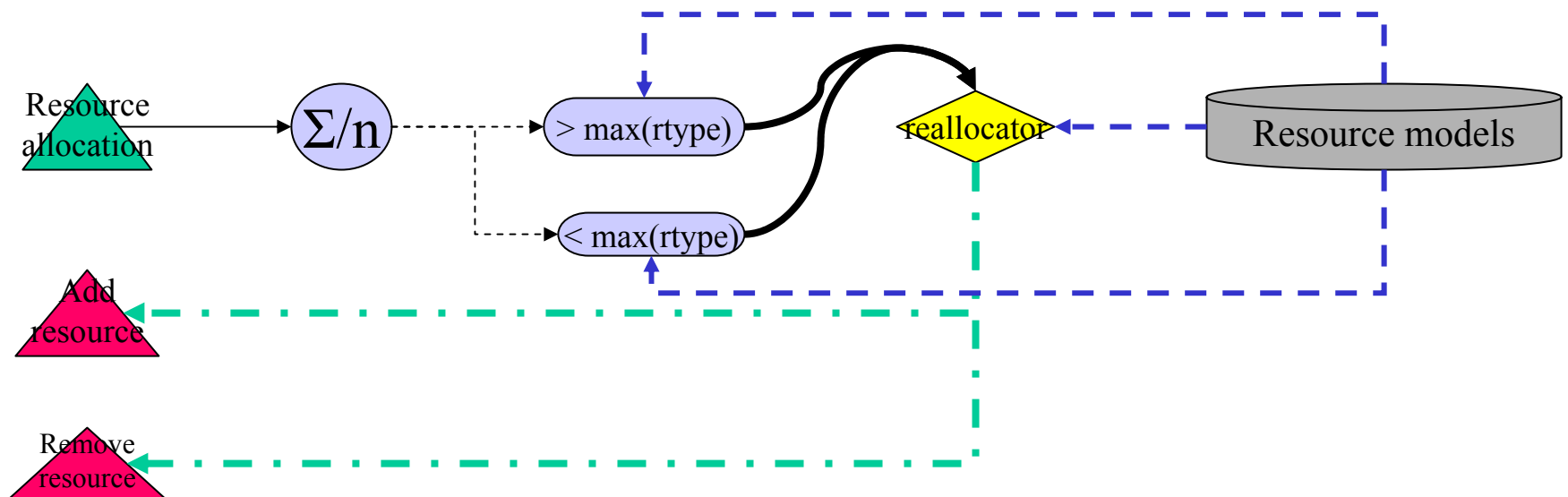
Self-Awareness Architectural Style

Patterns

- Examples
 - Resource allocation
 - Corruption resiliency
 - User authorization
 - Model Comparator
- Abstraction
 - Progress
- Composition
 - Authorization (revisited)

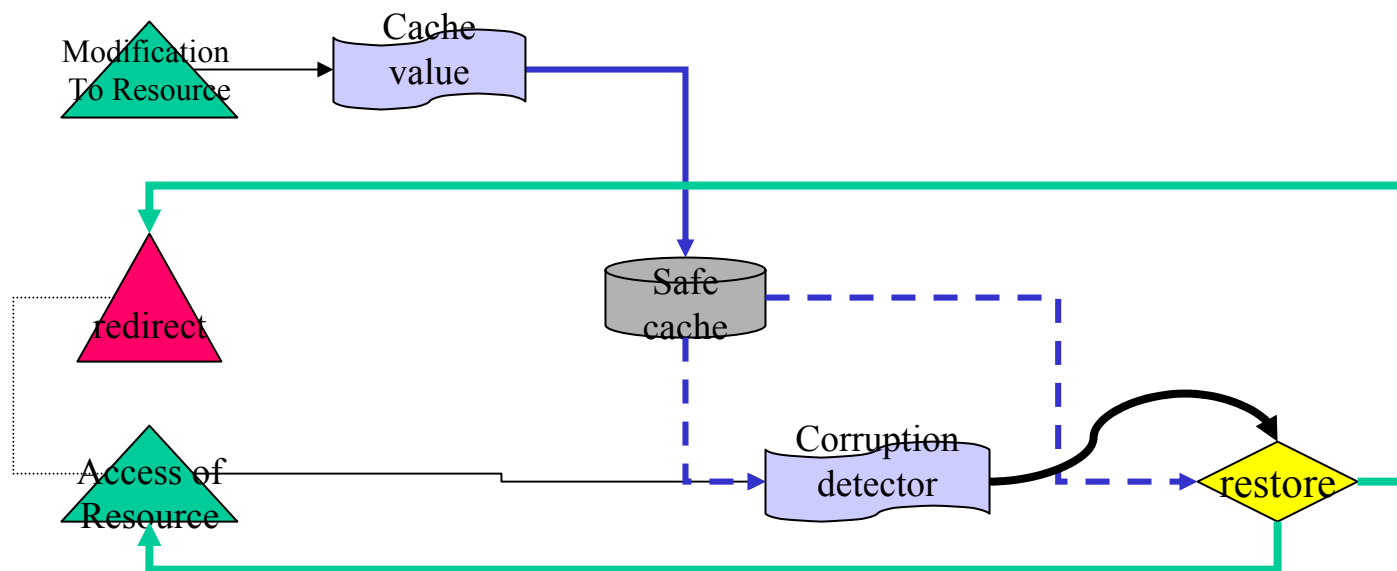
Resource Allocation

- probes watch resource consumption (allocation / deallocation)
- gauges determine average usage, looking for threshold violations.
 - These gauges may need to refer to some model for resource consumption;
 - for example, the thresholds may depend on the type of job being run.
- some decision logic determines how to reallocate resources,
 - either by adding new resources to one process or
 - removing resources already allocated to others.



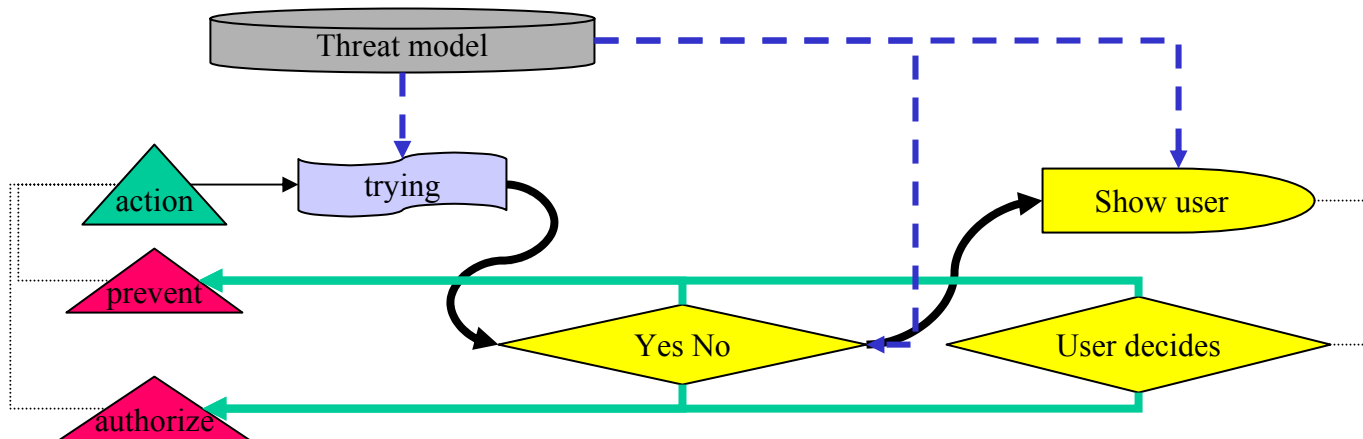
Corruption Resiliency

- probes into the system that capture all safe modifications to a resource.
- (Presumably, there are also paths in the system that allow unsafe modifications)
- gauge caches the safe modifications redundantly, but almost certainly more slowly
- corruption detector
 - e.g. during the access by computing a hash code on the real store and comparing it with a cache's code
 - the proper answers can be returned (access is redirected to the cache)



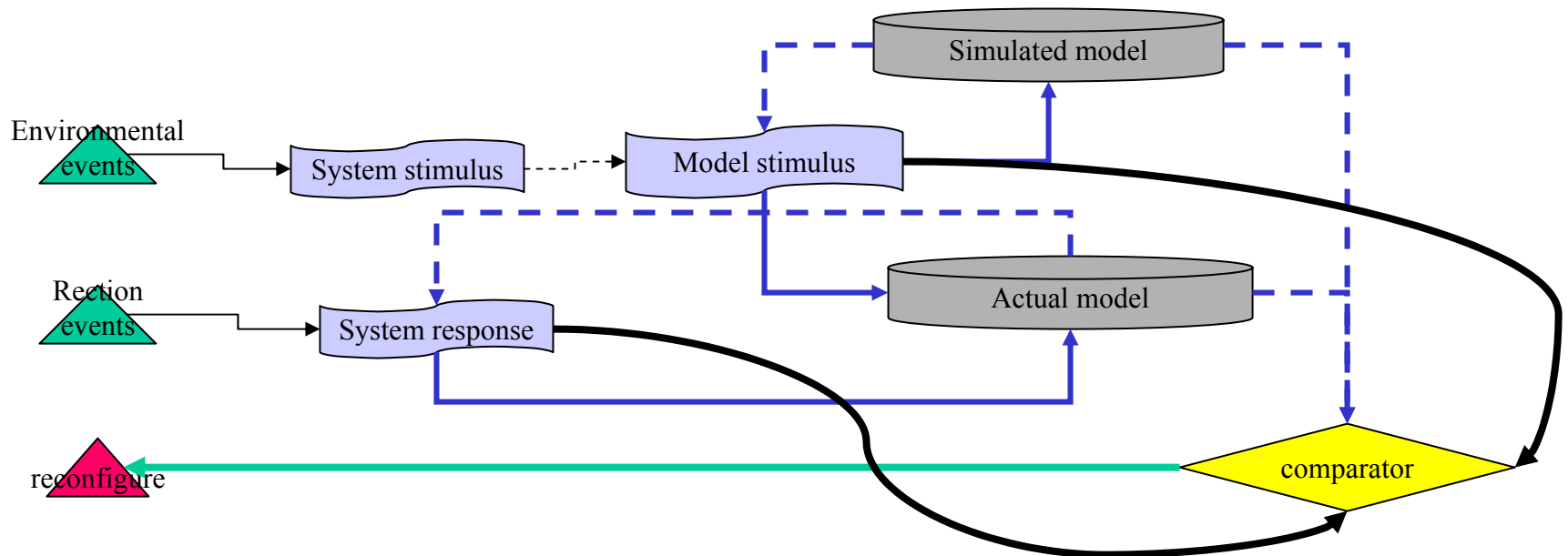
User Authorization

- of the managed system to allow questionable activities to proceed or not.
- gauge (“trying”) determines that a particular action is being attempted
- threat model is consulted and a decision is made on whether the action should be prevented or allowed to proceed
- if the decision cannot be made automatically, the user is informed via a display.
- user indicates the decision (by keyboard, mouse click, or timeout, perhaps) and effects the appropriate system response.



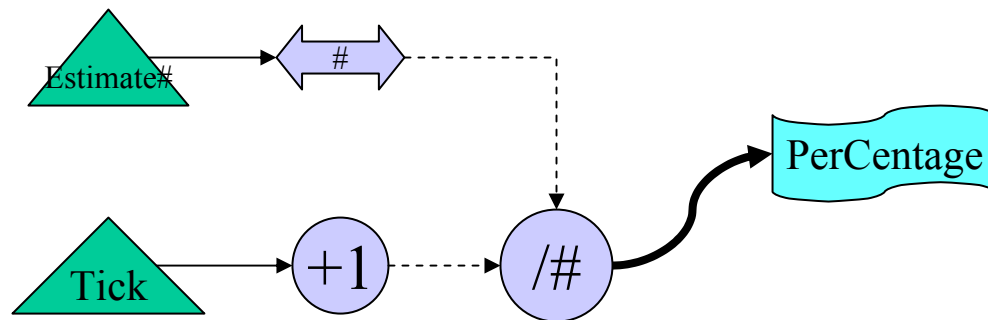
Model Comparator

- construct two somewhat independent models of a system
 - environmental events which drive the system collected by probes
 - E.g., an event such as “request print”
- a simulation, proceeds to determine a model response, building up the Simulated Model.
 - E.g., a finite state machine model may change from “allow requests” to “request pending.”
- system responses (from a set of probes) produce the “Actual Model”
 - E.g., the response probe might report that the system changed states to “printing.”
- comparator gauge determines whether a difference exists and the appropriate action is decided upon
 - E.g., the test would be whether “printing” and “request pending” are equivalent states.



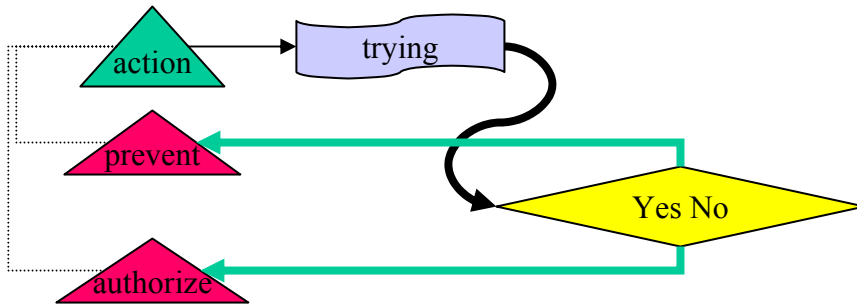
Towards Abstraction

- measurement event first announces the size of a set of items to be processed.
- each time an item has been processed, a “Tick” event is reported by the instrumented system
- a counter is increased.
- final gauge divides the counter value by the size of the set after each tick, thus dynamically indicating the percentage of the job that has been accomplished. .

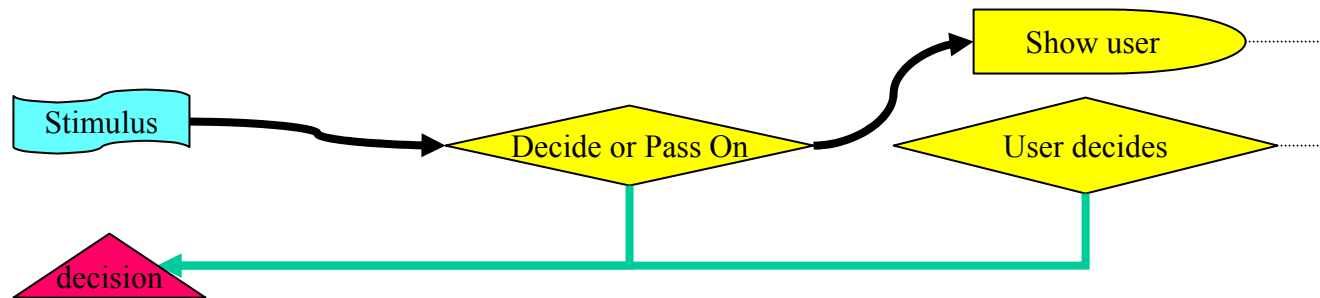


Composing Patterns

- User Intervention is more general in that the user is “deciding,” not saying “Yes or No.” (A way of bundling a group of connectors and components is needed.)
- To compose User Authorization from Authorization and User Intervention requires impedance matching the “decision” events to become a “yes and no.”
- One must be specific about where patterns can be introduced.
- Compound authorization requires generalizing Authorization.



Authorization



User Intervention

Issues

- Formalizing
- Openness / closedness of patterns
- Semantics
- Goal: codify knowledge in the self-management area
 - Categorize talks here
 - Any new ones seen?