



Hawaii International Conference on System Sciences
Hilton Waikoloa Village - Island of Hawaii
January - 2005

A Run-Time Adaptable Persistency Service using the SMART Framework

João W. Cangussu
www.cangussu.com

Kendra Cooper Eric Wong Xiao Ma



Department of Computer Science
The University of Texas at Dallas

1

Outline

- ◆ Motivation and Objectives
- ◆ The SMART Framework Architecture
- ◆ System Identification
- ◆ Fuzzy Logic
- ◆ Results
- ◆ Related Work
- ◆ Concluding Remarks



Department of Computer Science
The University of Texas at Dallas

2

Motivation

- ◆ Heterogeneous platforms with distinct resource constraints
- ◆ How to design software to cope with such complex environments
- ◆ Resource allocation is not always a solution
- ◆ Switch between alternative, though functionally equivalent, components



Objectives

- ◆ Allow the design of persistency adaptive systems with multiple solutions based on resource constraints
- ◆ Switch is done at run time with no recompilation
 - ◆ Insertion of a new component at run time depends on the use of a dynamic programming language
- ◆ Non-functional requirements of the components are “rigorously” specified

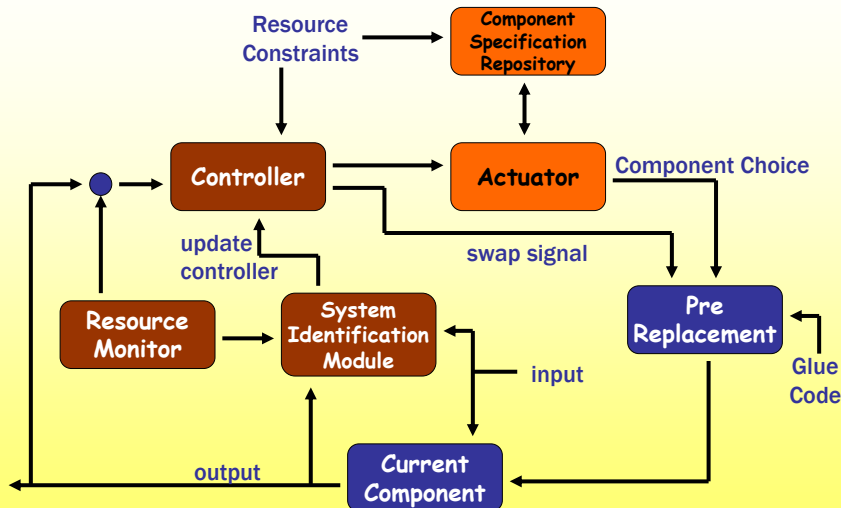


Main Goal

Allow the design of runtime adaptive systems coping with resource constraints in constantly changing environments.



The SMART Architecture



System Identification

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases}$$

$$Y(t) = \begin{bmatrix} x(t+1) \\ y(t) \end{bmatrix} \quad \Theta = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \quad \Phi(t) = \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}$$
$$Y(t) = \Theta \times \Phi(t)$$
$$\hat{\Theta}_N^{LS} = \left[\frac{1}{N} \sum_{t=1}^N \Phi(t) \Phi^T(t) \right]^{-1} \frac{1}{N} \sum_{t=1}^N \Phi(t) Y^T(t)$$

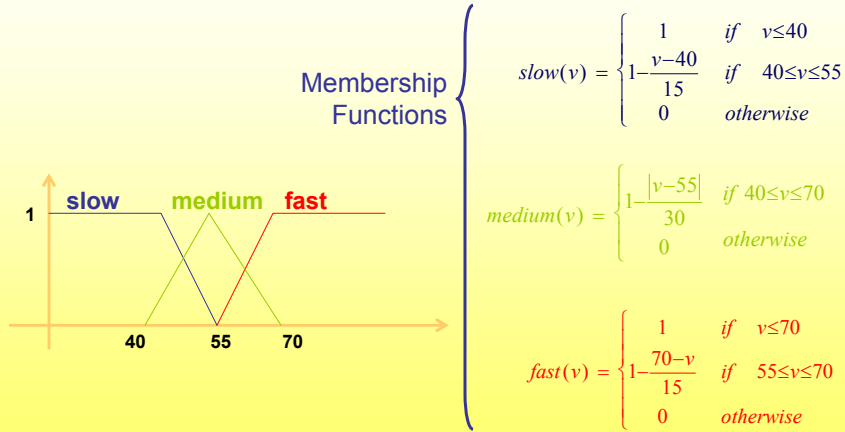


Fuzzy Logic

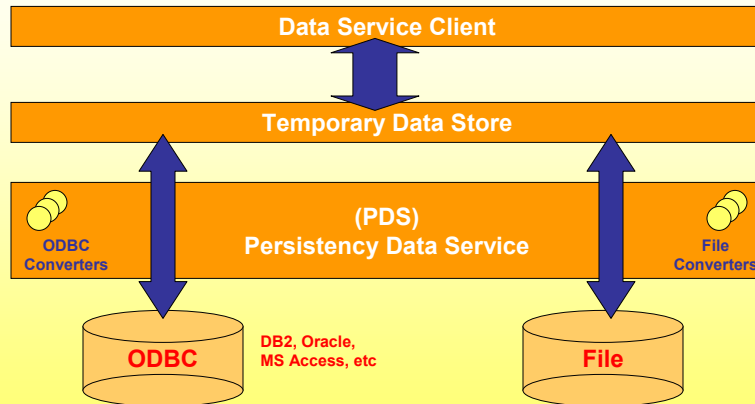
- ◆ Fuzzy logic has been chosen to be used in the SMART Framework:
 - ◆ Suitable for systems with mathematical models that are difficult to derive
 - ◆ Allows decision making with estimated values under incomplete or uncertain information
- ◆ Fuzzy multi-criteria decision making algorithms are available



Fuzzy Partitions



SMART Application: A Persistency Data Service



Persistence Components

- ◆ P_1 : uses an algorithm which saves an entity immediately after modifying it.
- ◆ P_2 : use an algorithms that saves all modified entities in memory to disk periodically



Components Solution Spaces

- | | |
|-----------------------|---------------------------|
| ◆ P_1 : | ◆ P_2 : |
| ◆ low memory usage | ◆ High memory usage |
| ◆ Slow response time | ◆ Fast response time |
| ◆ High data integrity | ◆ Moderate data integrity |

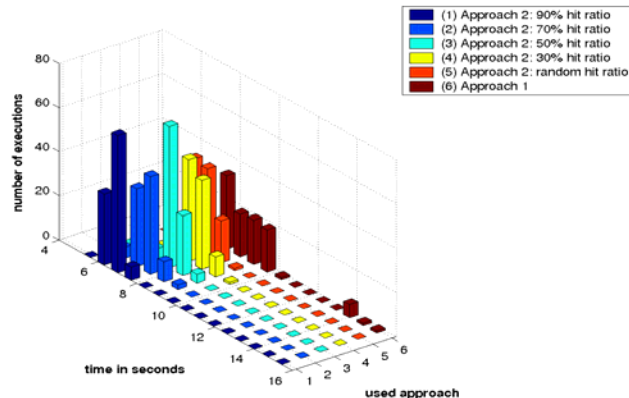


Component Evaluation

- ◆ Non-functional behavior measured at run time
- ◆ Each component is executed 100 times with 200 client request modifications each time
- ◆ P_2 is also considered for different hit ratios:
 - ◆ Fixed hit ratios ranging from 0.3 to 0.9
 - ◆ Uniform randomly distributed hit ratio



Component Evaluation Execution Time

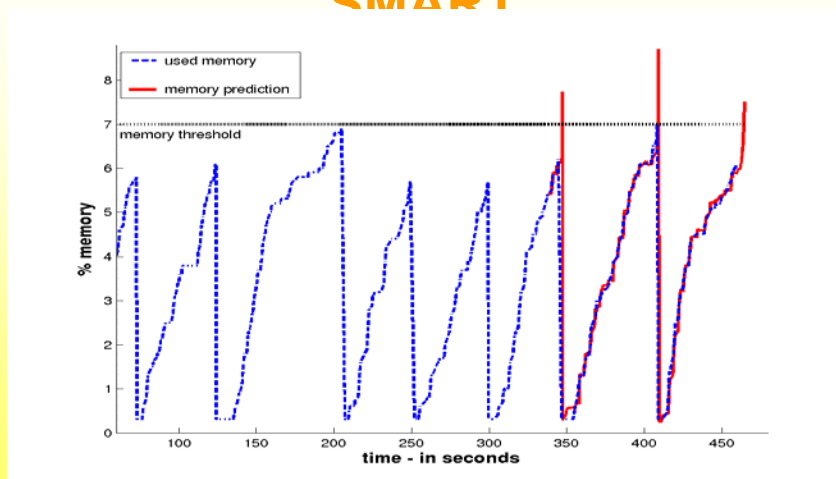


Adaptive Application

- ◆ Swap from P_2 to P_1 occurs once a memory usage of 7% or more is predicted
- ◆ Swap from P_1 to P_2 occurs when more than 5% of memory is available in the system



Snapshot of the Data Persistency Service using SMART

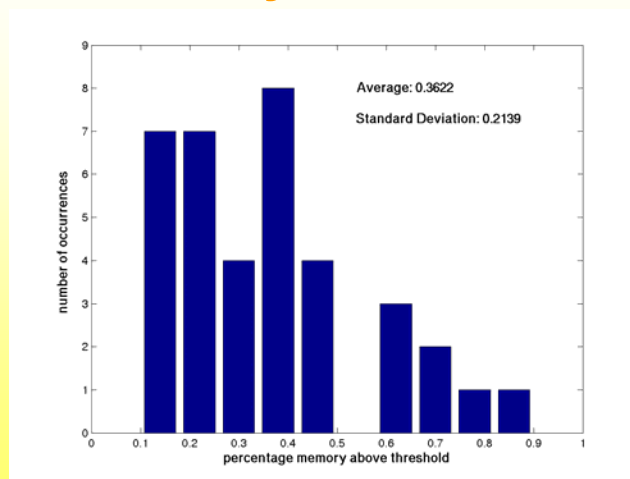


Data Persistency Service Results

- ◆ The Data Persistency service and the associated MATLAB scripts were executed over a period of 8.601 hours
- ◆ A total of 1812 swaps (906 from P_2 to P_1 and 906 from P_1 to P_2) were observed.
- ◆ Only 37 swaps were not execute in time (4.08% of the swaps from P_2 to P_1)



Data Persistency Service Results Memory Violations



Reasoning About the Failures

- ◆ Failures are not due to wrong predictions
- ◆ In almost all the cases, failures are due to late reaction time because:
 - ◆ Overhead in the prediction
 - ◆ Communication overhead between C and MatLab
 - ◆ Scheduling of the “Actuator” thread

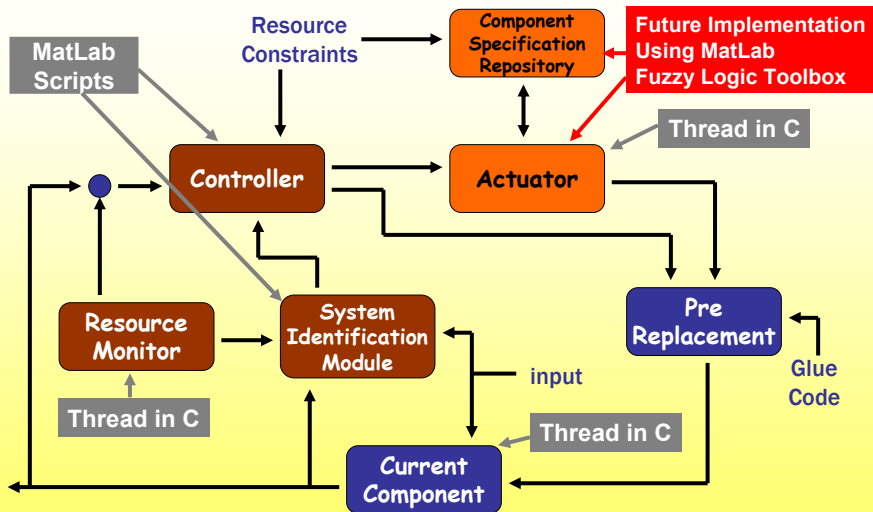


Potential Solution

- ◆ The overhead in the prediction could be minimized by:
 - ◆ having a compiled version of the MatLab scripts
 - ◆ Unfortunately, not all functions can be converted to C code.
 - ◆ Increasing the number of steps ahead use in the predictions
 - ◆ This would incur accuracy side effects
 - ◆ Using less data points in the prediction
 - ◆ This would also incur accuracy side effects



SMART: Prototype



Related Work

- ◆ Adaptive Systems
 - ◆ Containment Units
 - ◆ QoS Control Framework
 - ◆ nitro Reflective platform

Concluding Remarks

- ◆ Adaptive environment based on the sound theory of feedback control and fuzzy logic
- ◆ SMART is not domain specific
- ◆ Two major advantages
 - ◆ Predictability
 - ◆ Flexibility
- ◆ Encouraging results
 - ◆ Avoided 95% of constraint violations



Future Work

- ◆ Overhead decrease
- ◆ Further investigation of system identification techniques
- ◆ Implementation and integration of the Fuzzy multi-criteria decision making algorithm for component selection and specification
- ◆ Validation with additional applications
 - ◆ Data compression
 - ◆ Image interpolation
 - ◆ Digital Filters



Potential Solutions

- ◆ **Interprocess communication can be use to minimize:**
 - ◆ **Scheduling problems**
 - ◆ **Communication overhead between C and MatLab**



Programming Language Issues

- ◆ **Use of Java allows insertion of new components at run-time**
 - ◆ **Garbage collector causes side-effects on memory usage**
- ◆ **C/C++ makes the environment more stable with respect to memory utilization**

