A Conflict Resolution Control Architecture for Self-Adaptive Software

Prof. A. Taleb-Bendiab

School of Computing and Mathematical Sciences

Liverpool John Moores University

{cmsnbadr, d.reilly, a.talebbendiab}@livjm.ac.uk

http://www.cms.livjm.ac.uk/except

Dependable software

- Autonomic computing: a recent trend
  - Devolving software management, maintenance to software
    - Self-organising, self-healing, sentient, self-adaptive, self-aware, etc.
  - Requiring meta-systems and meta-reasoning to;
    - Continuous measurement and/or reflection on operational systems
- High-assurance: high-{integrity, availability, etc.}
  - Complexity and uncertainty hiding through;
    - adaptive capability to respond to changes including: fault&intrusion-tolerance, thus masking errors, failures, etc.
    - Dynamic architecture transformation and reconfiguration strategy;
      - This requires reasoning and consideration of a set of concerns;
        - » software architecture model including; components and their interactions, the properties and policies,
        - » Style and composition rules and/or norms that limit the allowable systems adaptation operations.

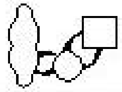
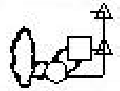
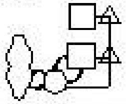
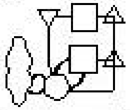
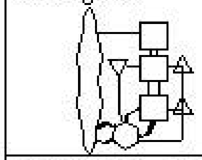
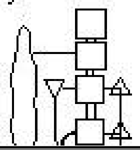
Integrity Management

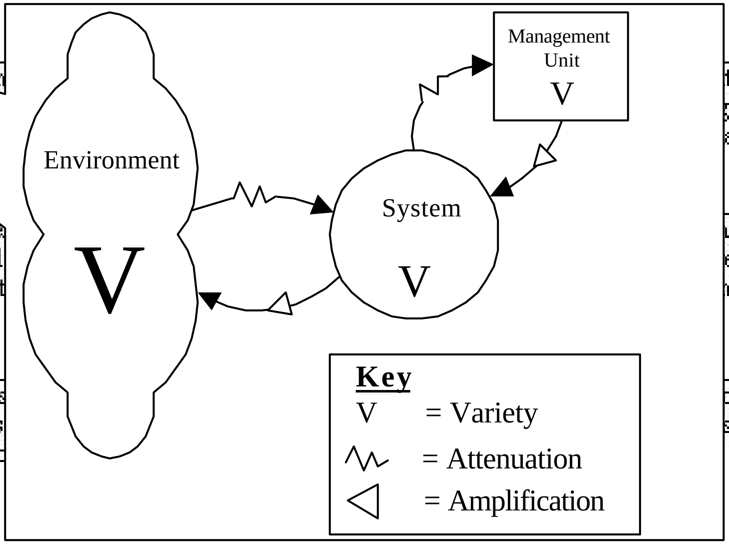
- Dynamic architecture transformation often lead to inconsistencies and conflicts
  - Systems integrity
  - Quality of service, etc.
- Requirement for a software adaptation engine with;
  - Conflict detection and identification
  - Conflict resolution
    - Solution generation, negotiation
    - Change plan enactment, etc.
  - Control strategies defining;
    - Transformation rules, regulations, patterns, etc.
- Our approach is a middleware to support for self-adaptive software conflict management.

Related Work

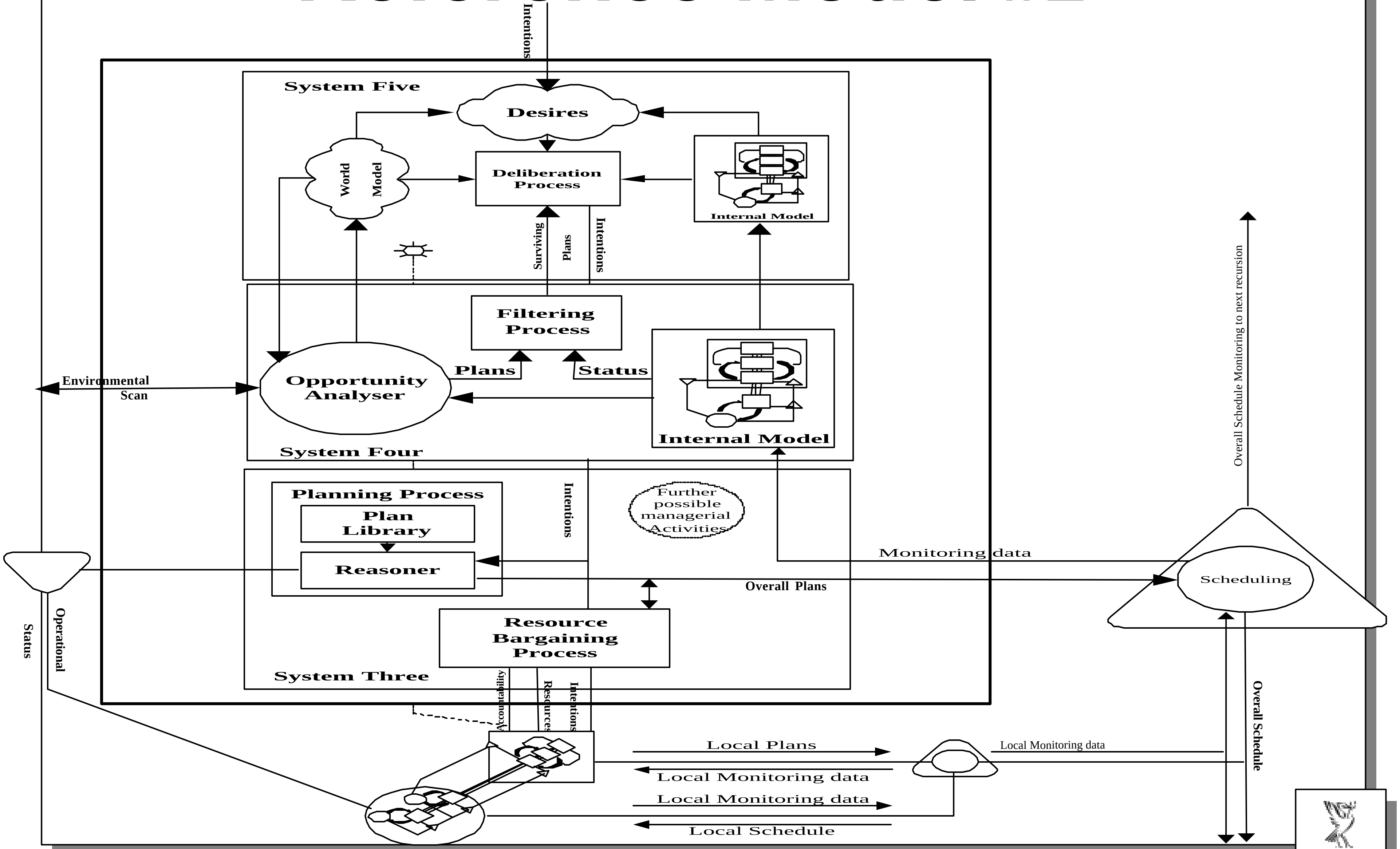
- Self-Adaptive Software
  - Can be defined as software with computational reasoning capabilities to monitor and change its own structure and/or behaviour to adapt to its operating environment and recover from errors.
- Reflective middleware
- Dynamic configuration control and management
- Conflict resolution
  - Negotiation Protocol
  - Exception handling

Reference Model #1

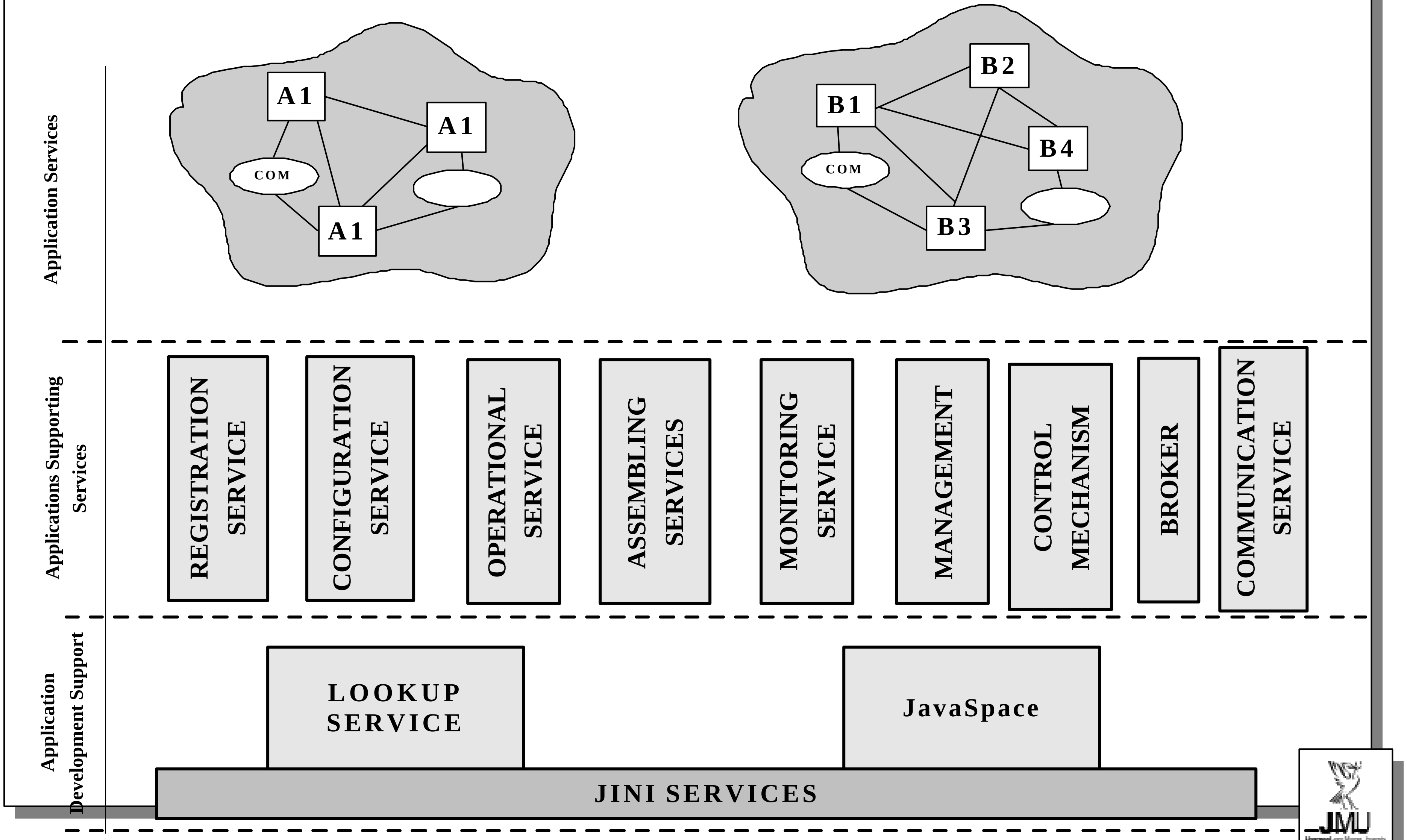
System Identifier	System Type
System One (S1) Operations 	System One performs the productive operations of the organization. An organization may be composed of a number of S1s, each providing a distinct product or service. Each S1 consists of an operational element controlled by a management process and in contact with the operational environment and is in some respects is similar to the plant/management arrangement adopted by control system theory.
System Two (S2) Coordination 	Sys concern tory in that it attempts to contain or minimi facilities such as scheduling and standardiz al S1 needs.
System Three (S3) Control 	System Three is conce e management processes contained within this system will esource provision and strategic plan production, although st han in the normally accepted sense.
System Three* (S3*) Audit 	System Three* (read as ogress and provides direct access to the physical operations s essentially provides additional data over and above that pro
System Four (S4) Intelligence 	System Four is concerned with planning the way ahead in the light of external environmental changes and internal organizational capabilities. To this end, S4 'scans' the environment for trends that may be either beneficial or detrimental to the organization and constructs developmental organizational plans accordingly. To ensure that such plans are grounded in an accurate appreciation of the current organization, the intelligence function contains an up-to-date model of organizational capability.
System Five (S5) Policy 	System Five determines the overall purpose of the organization i.e. defines the activities that are performed by S1s. As such S5 represents the policy-formulation or normative planning function. Policy formulation is informed by a "world-view" provided by S4 and representing the current beliefs and assumptions held by the system about the environment and models of current organizational capability populated by data flowing from the lower level systems in the organization.



Reference Model #2



An Architectural Model



Application Services

Applications Supporting Services

REGISTRATION SERVICE

CONFIGURATION SERVICE

OPERATIONAL SERVICE

ASSEMBLING SERVICES

MONITORING SERVICE

MANAGEMENT

CONTROL MECHANISM

BROKER

COMMUNICATION SERVICE

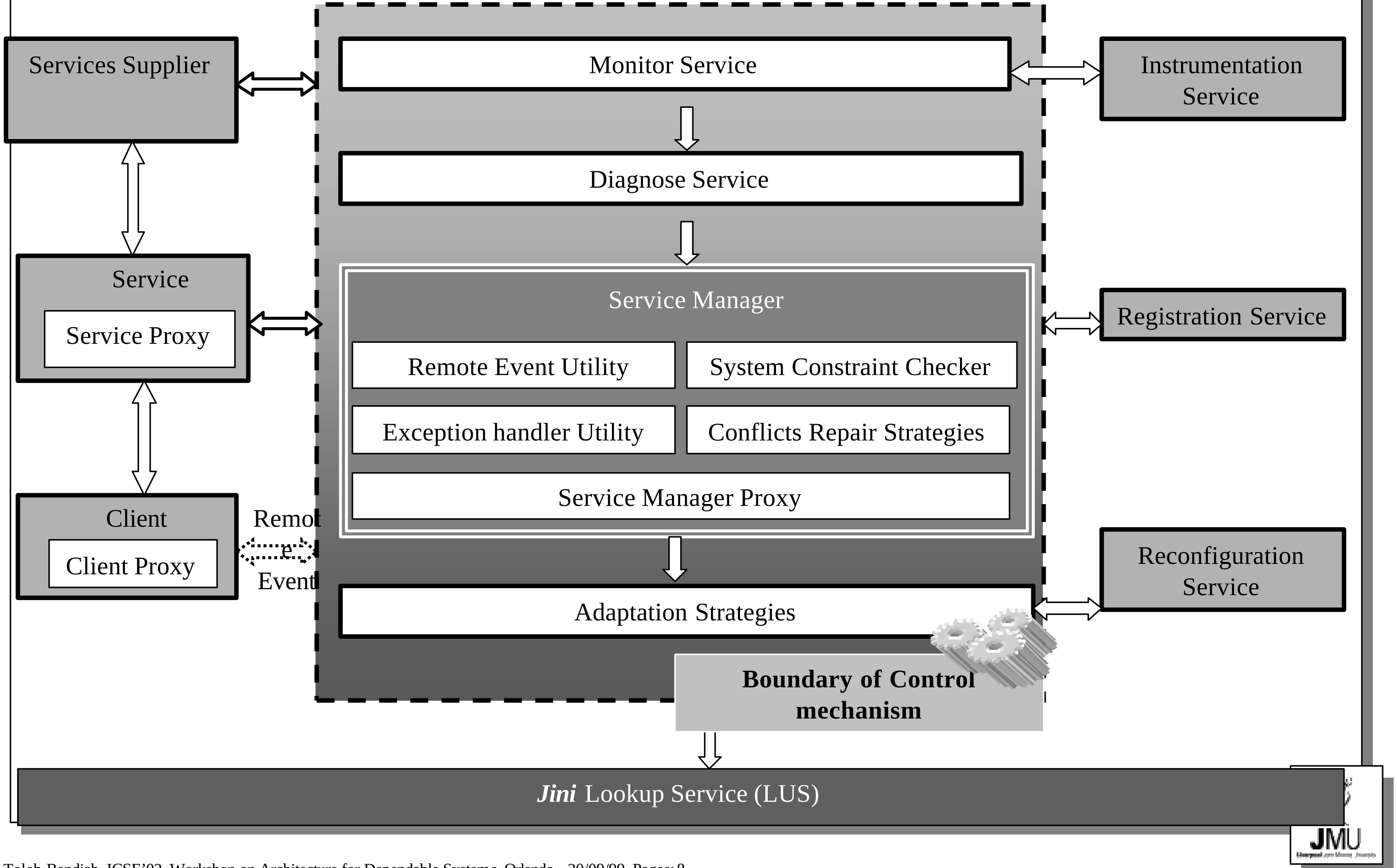
Application Development Support

LOOKUP SERVICE

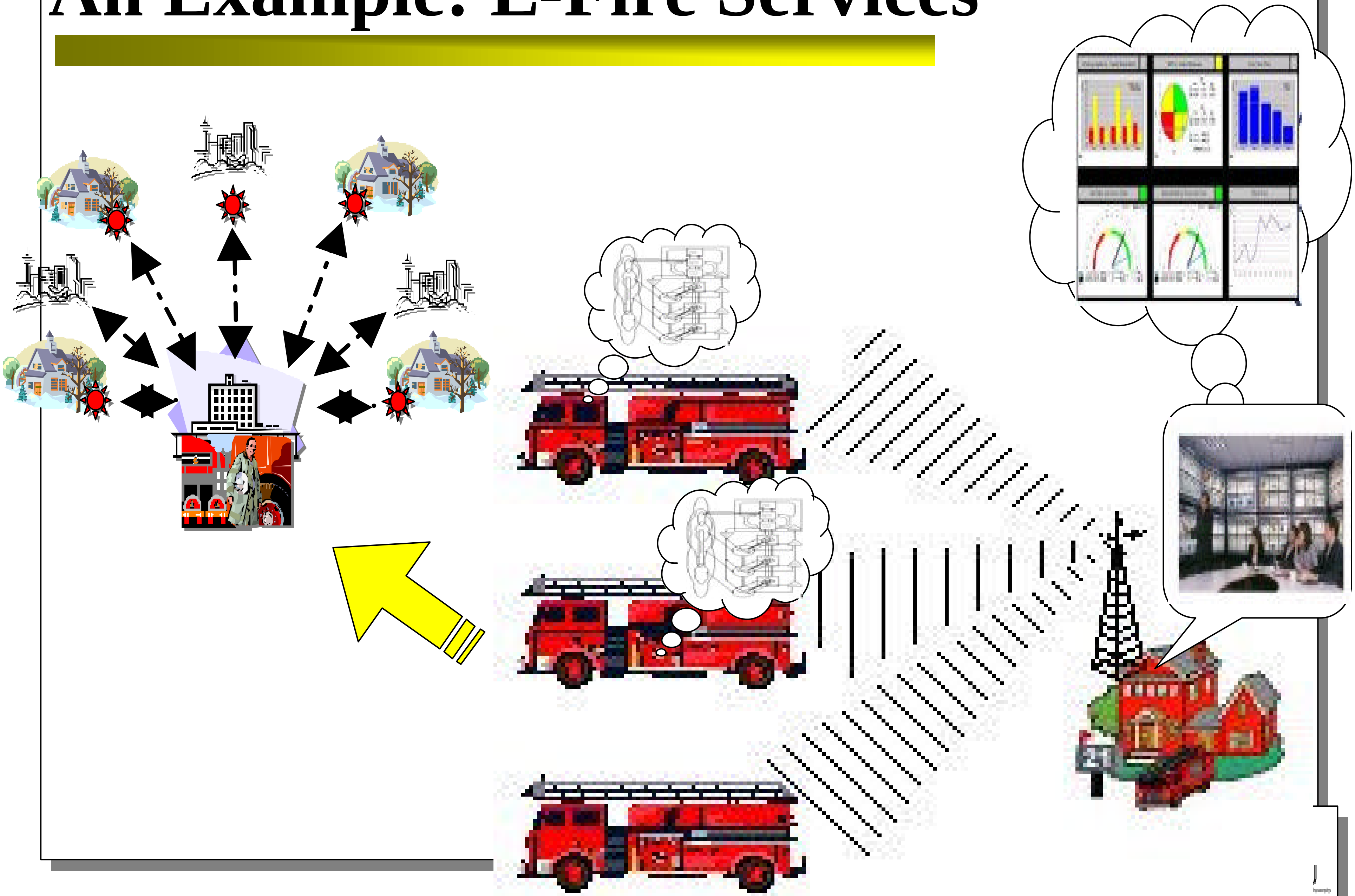
JavaSpace

JINI SERVICES

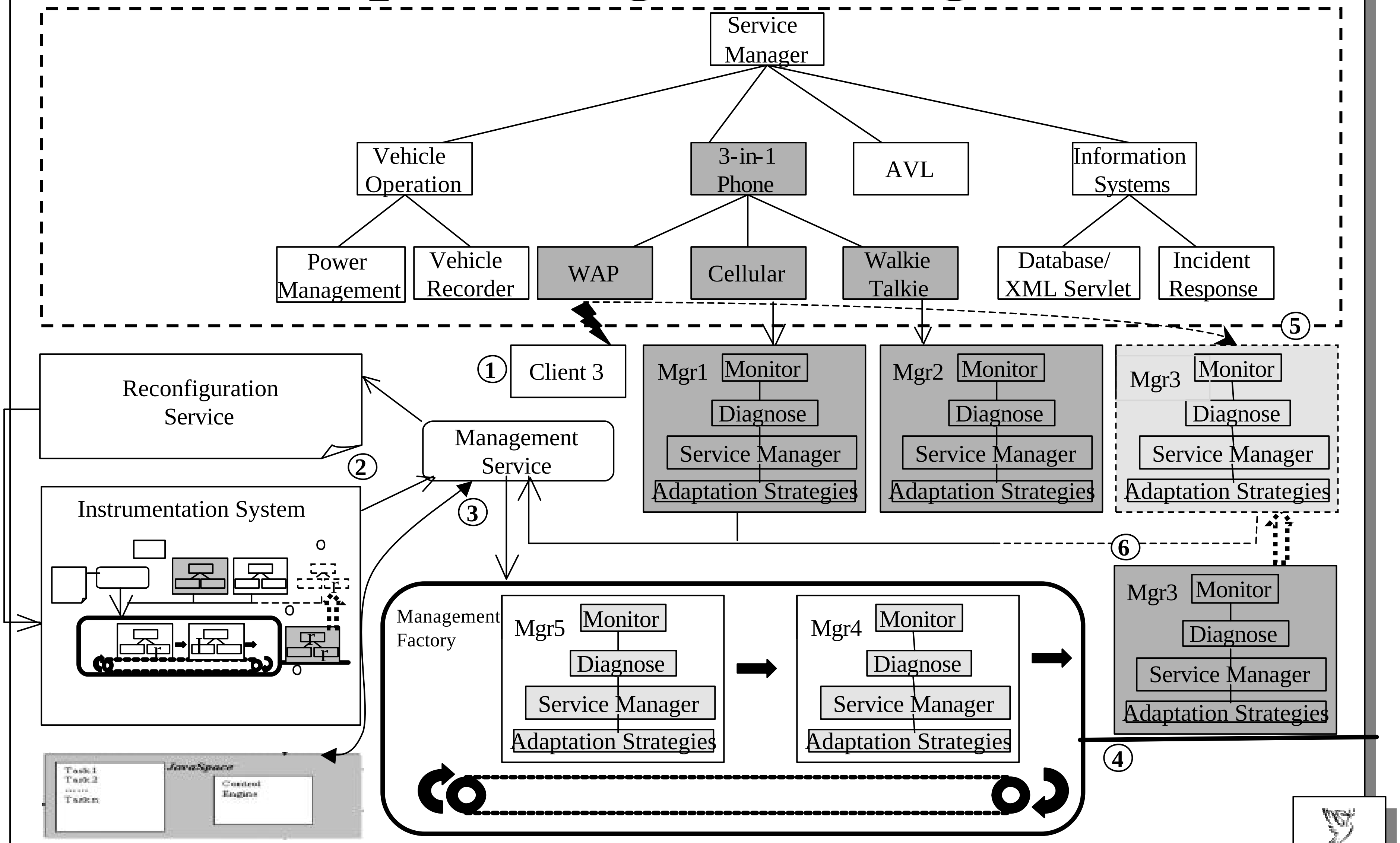
Adaptation Usage Model



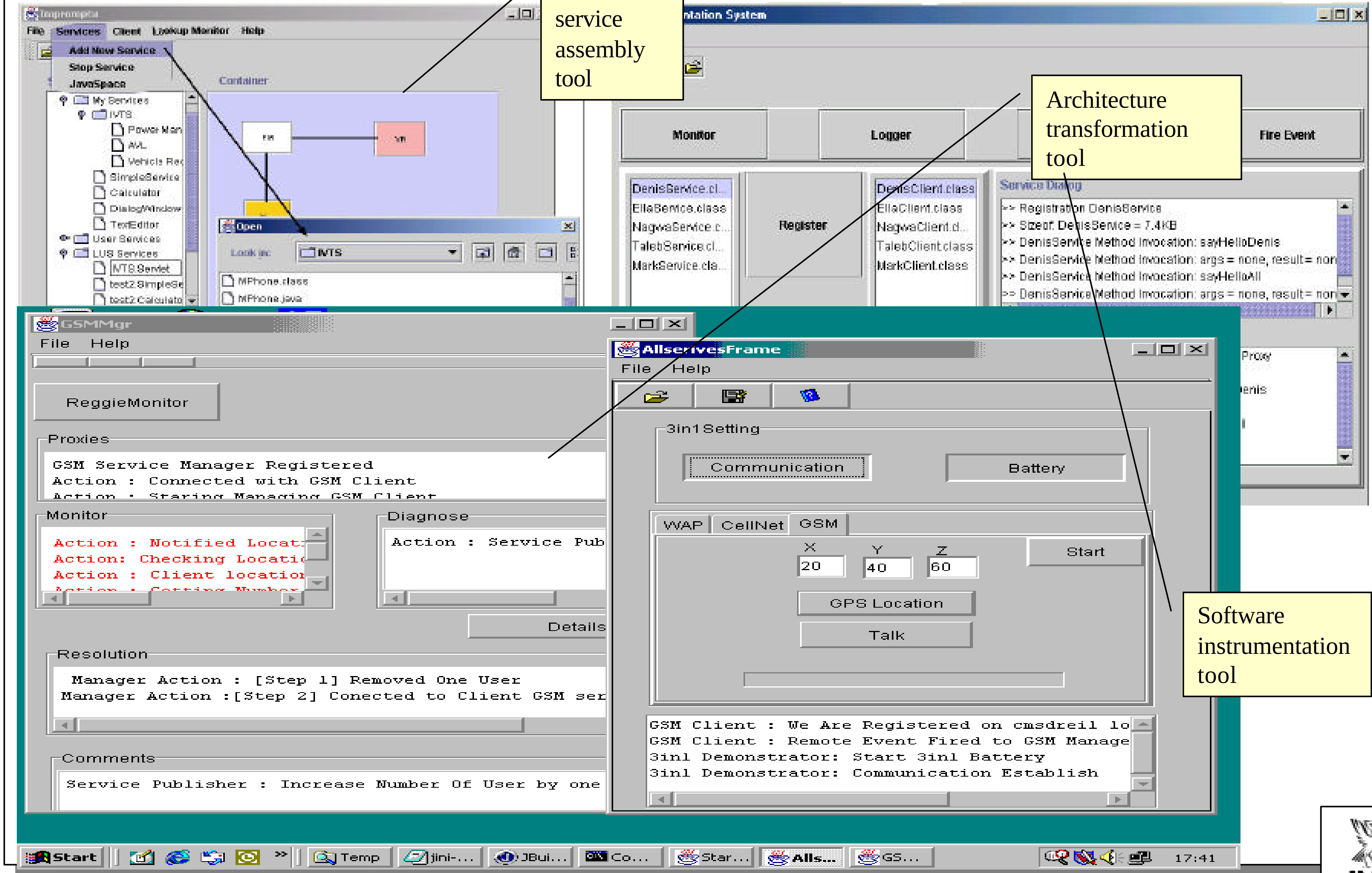
An Example: E-Fire Services



Example Programming Model



Demonstrator



Ad-hoc service assembly tool

Architecture transformation tool

Software instrumentation tool

```
DenisService.cl...
EllaService.class
NagwaService.c...
TalebService.cl...
MarkService.cla...
```

```
DenisClient.class
EllaClient.class
NagwaClient.cl...
TalebClient.class
MarkClient.class
```

```
>> Registration DenisService
>> Sizeof DenisService = 7.4KB
>> DenisService Method Invocation: sayHelloDenis
>> DenisService Method Invocation: args = none, result= non
>> DenisService Method Invocation: sayHelloAll
>> DenisService Method Invocation: args = none, result= non
```

Proxies

```
GSM Service Manager Registered
Action : Connected with GSM Client
Action : Staring Managing GSM Client
```

Monitor

```
Action : Notified Locat
Action: Checking Locatio
Action : Client locatio
Action : Getting Number
```

Diagnose

```
Action : Service Pub
```

Resolution

```
 Manager Action : [Step 1] Removed One User
Manager Action :[Step 2] Conected to Client GSM ser
```

Comments

```
Service Publisher : Increase Number Of User by one
```

X: 20 Y: 40 Z: 60

```
GSM Client : We Are Registered on cmsdreil lo
GSM Client : Remote Event Fired to GSM Manage
3in1 Demonstrator: Start 3in1 Battery
3in1 Demonstrator: Communication Establish
```

Conclusions & Future Work

- Presented an architecture for conflict resolution and management for
  - Self-adaptive software
  - Supplied as a middleware service
- Presented an example illustrating;
  - Propose programming model
  - Usage model
- Further work
  - Resolution session control and management
  - Evaluation.

```
        ``
      .0O1.^
      u$ON=1
      z00BA|
     |..=~.
    ;s<'''
    NRX~=-`
    z0c^<X^
    ~B0s~^`
     @@$H~'
    n$0=XN;.`
   iBBB0vU1=~'`
   `$@00cRr`vul
    FAHZuqr-'
    ZZUFA@F|.`
   ;BRHv n$U^-
  `ARN1   ^@si
 'Onv~     O1.'
 cOqr      rs.`
 aUU`       ul`
`RO-         :.`
nn~`         -=.~|-`
=1^'..`      `..`
```

That’s the end – so I’m off !