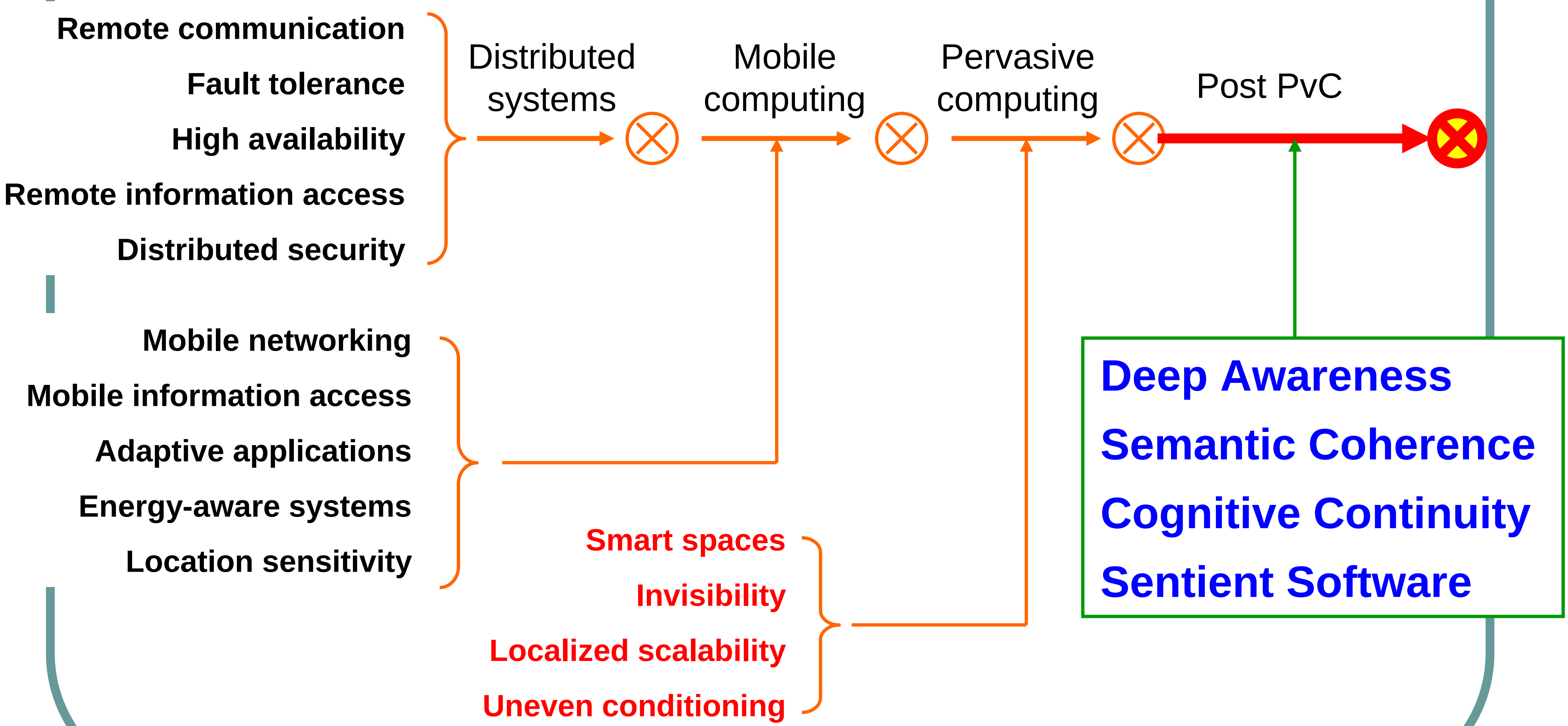
HKU Sparkle Project

Dr. Cho-Li Wang

The University of Hong Kong

Aug. 4, 2006.

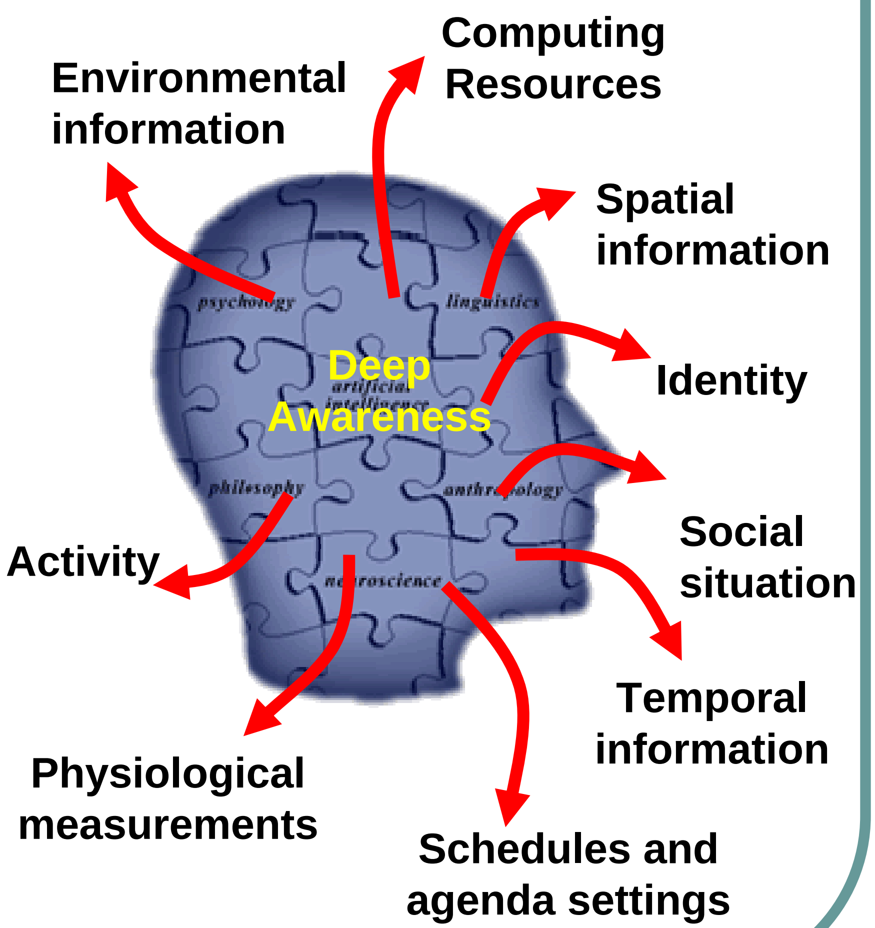
The Pervasive Expedition



"Pervasive Computing: Vision and Challenges"
M. Satyanarayanan [CMU, Aura Project, 2001]

1. Deep Awareness

- The majority of context-aware computing to date has been restricted to ***location-aware computing*** for mobile applications (location-based services).
- ***Deep Awareness***:
  - Make full use of context information
  - Make use of "commodity sensors" (e.g., WebCam, RFID, Temp/Light,..)

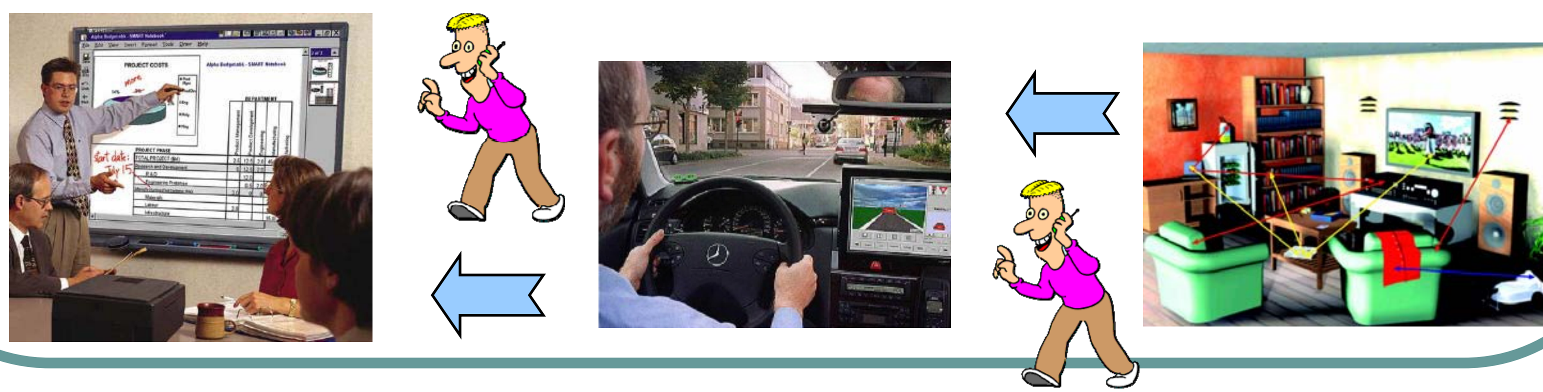


2. Semantic Coherence

- **Contextual Level Interoperability**
  - Substitutability : if a service could be substituted by another one
- **Mechanism : Runtime Ontology Mapping:**
  - Normally each smart space has its own ontology or knowledge base
  - In an open environment (instead of a closed smart space), it is not practical to assume a unified ontology.
  - A runtime ontology mapping mechanism is needed

Semantic Coherence

- Our design goals:
  - Support lightweight ontology mapping for smart spaces interoperation **with only partial information/knowledge**
  - Flexible smart space infrastructure to accommodate all kinds of ontologies



3. Cognitive Continuity

- **High user mobility in Pervasive Computing Environment**
  - **Mobility may raise user distraction as he/she experiences new smart spaces**
- **Our Proposal:**
  - Proactive task state synthesizing
  - Mapping and infusing between different plans.

4. Sentient Software

- **Sentient Software**
  - Context changes → Run-time changes of software behavior.
  - ***Commodity AI***: make the software look smart some of the time (implement some decent adaptive heuristics)
- **Current software systems:**
  - never disagree with anything you say, and of course they never initiate anything.
- **Some thoughts:**
  - It is almost impossible to know what the user really wants.
  - Observed that most people live in routing life and most human tasks are predictable
  - So we just build software which conforms more closely to how they work.

4. Sentient Software

- **Focused Issue:**
  - Dynamic Configuration and Reconfiguration
    - To dynamically construct the IS according to user's computational intention and resource availability
    - The basic concept is based on dynamic composition techniques
      - ***Separation of concerns***
      - ***Component-based***
      - ***Computational Reflection***
- **Requirements**
  - User-centered Configuration
    - Configure in the user-preferred way
    - Activity Theory, Mental Model, Situation-based
  - *Utility-based* Reconfiguration
    - Change the resource availability, meanwhile guarantee the user's satisfaction
    - Being able to adapt to the dynamics of the environment at the rate at which the dynamics, the changes, occurred.

Sparkle Legendary

- Sparkle I – Functionality adaptation
- Sparkle II – Semantic adaptation
- Sparkle III – Deep awareness

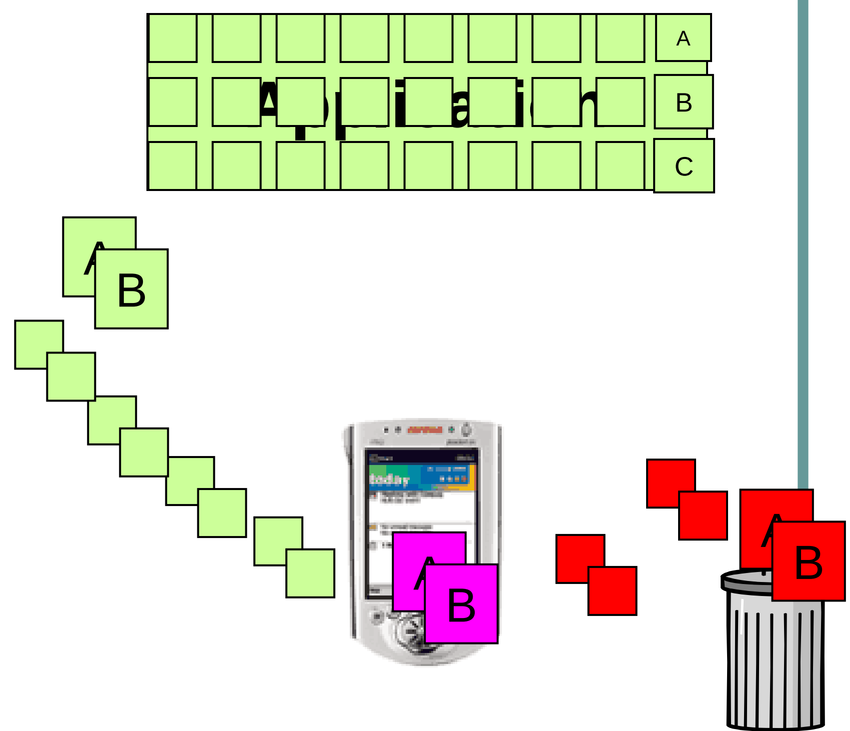
Sparkle I – Functionality adaptation

- Early Works of Sparkle.
- A new component paradigm: ***Facet Model***
- ***Separation of code and data***, preparing for
  - ***Migration***: State is kept in container
  - ***Adaptation***: code and data can be adapted individually
- ***Functionality Adaptation***
  - Components of the same functionality have varied granularity and/or feature

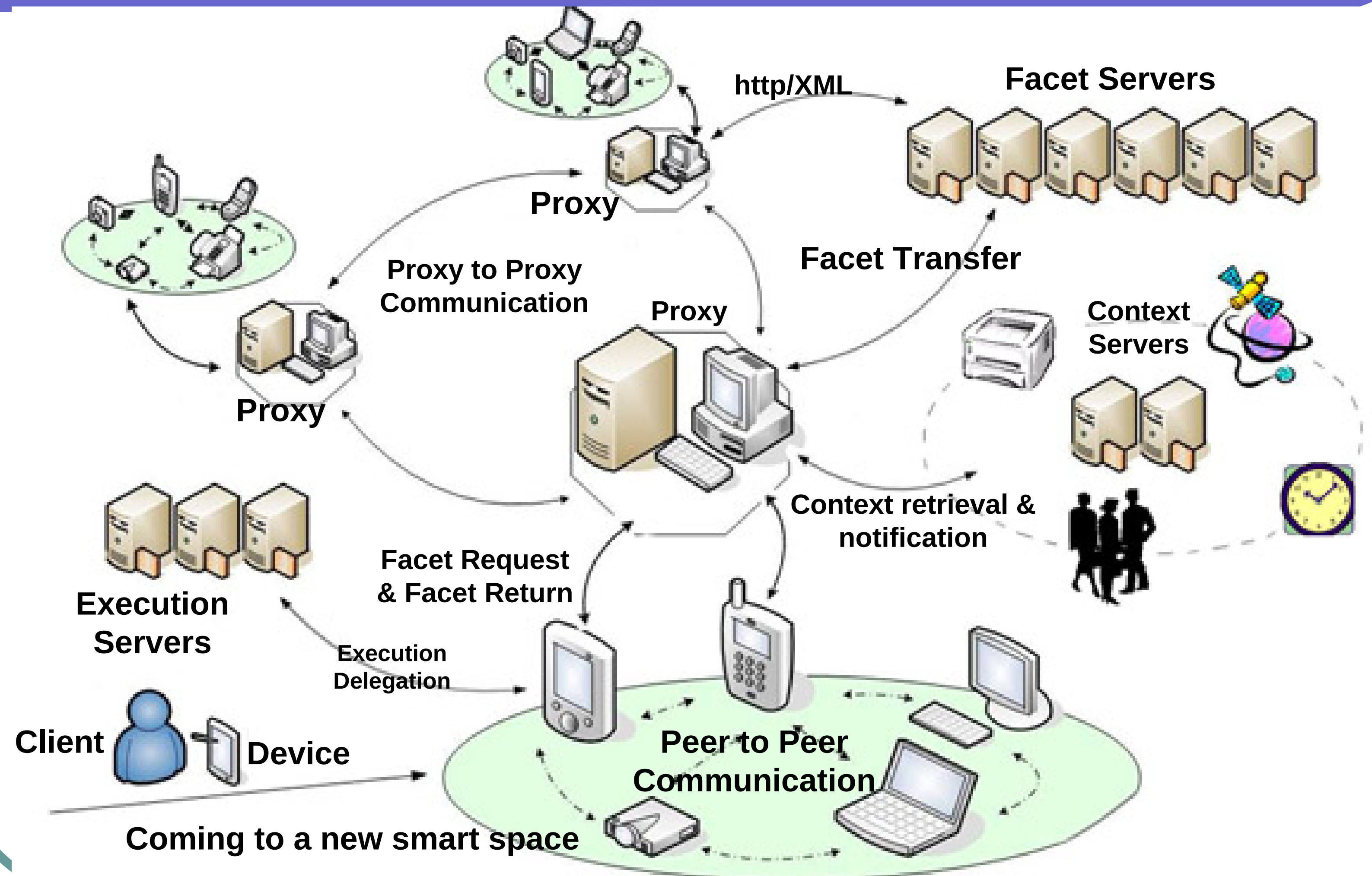
Sparkle I

- Dynamic Software Architecture for Pervasive Computing – "Computing in Small"
- Computing Anytime, Anywhere, at Any device, and support Any Application.
- Focus: Resource-awareness

Applications distributed as monolithic blocks



Sparkle I – Overview



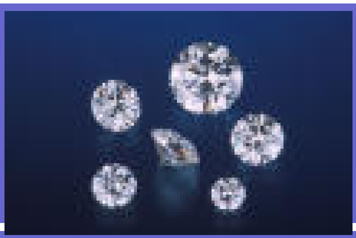
Facet Model

- Functionality
  - single well-defined task in an application
    - E.g. blurring an image, matrix multiplication
  - Given a set of inputs, it determines what changes are made and the outputs attained
  - **Contract** which specifies
    - Set of input & output parameters
    - Description of what is carried out
    - Pre-conditions and Post-conditions
    - Side effects
  - Identified by a funcID

Facet Model

- Facets
  - Pure functional units
  - Downloaded to client devices on demand
  - Can be cached in clients.
  - Implement single functionality
    - single publicly callable method
  - Stateless
    - Makes it throwable & replaceable at run-time

Facets



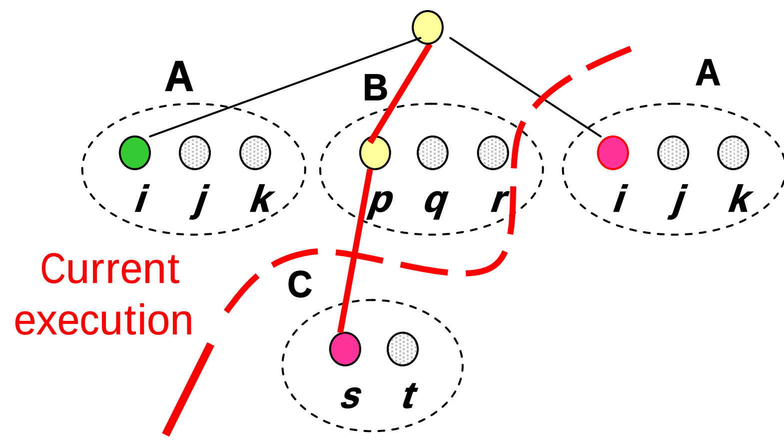
Facets – flat planes which make up a diamond

- **Shadow:** specifies properties of the facet
  - ***General info***: facetID, vendor, version
  - ***Functionality info***: funcID
  - Input and output specification, (data type/format…)
  - ***Resource requirements***: memory, processing, bandwidth, etc
  - ***Dependencies***: some other functionality it requires to finish its task etc.
  - Represented in XML format.
- **Code Segment**
  - **Executable code to achieve the functionality (written in Java)**
  - **Does not keep any permanent state**
- **A JAR file to box them together**

Facet Dependency Graph

Facet Dependency Graph

- Facets may call upon other facets to achieve their functionality
- May have more than one facet fulfilling the functionality (e.g., i,j, k for A)
- Dependency types:
  - “compulsory”
  - "optional" : "if-then-else“

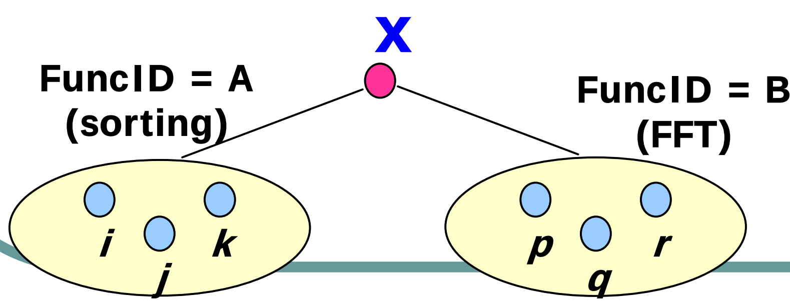


During execution, facets which are no longer active can be thrown

Inactive Facet
-already executed
completely

Facet which
Has not yet been
Brought in/loaded

Active Facet
- currently running



i: quick sort; i: bubble sort; k: merge sort

Shadow: Resource Requirement

- **Static resource requirements :**
  - **do not change at runtime**
  - **E.g., static data, program code,..**
- **Dynamic resource requirements**
  - **may change depending on various run-time conditions, such as size of the inputs, algorithm etc.**
  - **E.g., a blur facet depends on the size of the image**
- **Specified by the facet programmer (development time)**
  - **a formula, e.g., $3n^2+5m$, or**
  - **a look-up table, interpolation if required**
  - **only about the current facet**

Shadow: Resource Requirement

```
<memory>
  <static>233</static>
   <dynamic>
    <input_variables>
      <parameter name="m">
      <parameter name="n"> 2
    </input_variables>
    <formula> 3n^2+5m </for
   </dynamic>
 </memory>
```

```
<memory>
    <static>233</static>
    <dynamic>
      <input_variables>
        <parameter name="m"> 1 </parameter>
        <parameter name="n"> 2 </parameter>
      </input_variables>
      <table> <entry>
          <input name="m“< 20 </input>
          <input name="n“< 10 </input>
          <value> 400 </value>
        </entry> <entry>
          <input name="m“< 40 </input>
          <input name="n“< 30 </input>
          <value> 2900 </value>
        </entry>
        .
        .
      </table>
    </dynamic>  </memory>
```

Shadow: Examp

<memory>
<static>128</static> (in KB)
<dynamic>
<input_variables>
```
<parameter name="m"> 1 </parameter>
<parameter name="n"> 2 </parameter>
```
</input_variables>
<formula> 3n^2+5m </formula>

<identifier>GB00056</identifier>
<name>***IV.GaussianBlur***</name>
<vendor>SRG SANG</vendor>
<version>
<major>1.0
<minor>a</
</version>
<functionality

Pa

<dependencies>
<dependency order="1" type="optional" subtype="if-then-else">
<functionality_id>200016</functionality_id>
</dependency>
<dependency order="1" type="optional" subtype="if-then-else">
<functionality_id>200017</functionality_id>
</dependency>
<dependency order="1" type="optional" subtype="if-then-else">
<functionality_id>200018</functionality_id>
</dependency>
<dependency order="2" type="compulsory">
<functionality_id>200030</functionality_id>
</dependency>
</dependencies>

Part 3: Dependencies

Facet Request

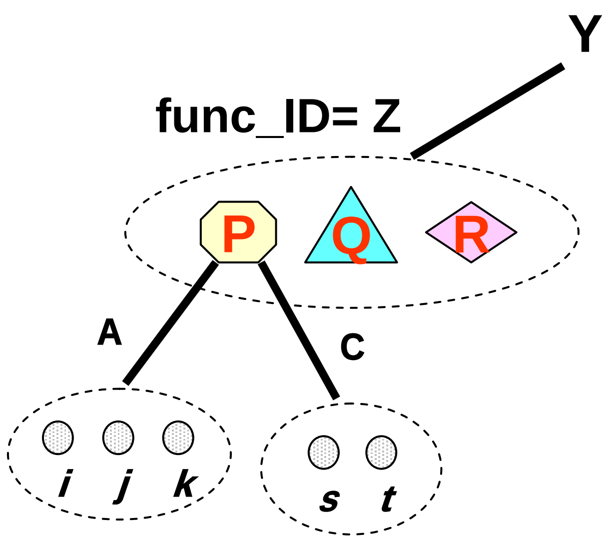
- Facet specification is sent to a proxy
  - Functionality, funcID, vendor, version,
  - Resource conditions (availability) in client
    - Memory, processing power, network conditions
  - The facet specification is changed into XML format and sent over SOAP.
- Proxy identifies a suitable facet (or several in a group) and sends it to the client
  - Match the criteria with the shadows of the facets available
  - Find a facet suitable to run under specified resource constraints
  - The proxy responds to the request by returning the matched facet(s) as a MIME attachment to a SOAP response.

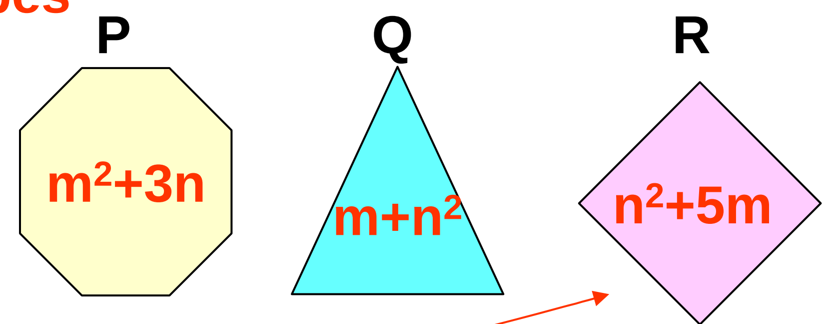
Facet Request

```xml
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body> <ns1:GetFacet xmlns:ns1="FacetProxy">
     <facet>
       <functionality_id>20003</functionality_id>
       <vendor>SRG SANG</vendor>
     </facet>
     <rootfacet>no</rootfacet>
     <context>
       <user> <identifier>vjwmkwan</identifier></user>
       <static_resource> ... </static_resource>
       <runtime_resource>... </runtime_resource>
     </context>
   </ns1:GetFacet></SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Dynamic Resource Requirement

- **The proxy compares the *resource requirement* of facets with the resource availability in the client.**
- **Proxy will send a facet whose resource requirement + the resource requirements of all its dependencies together is less than the resource availability in client.**
- **Consider the dependency types**





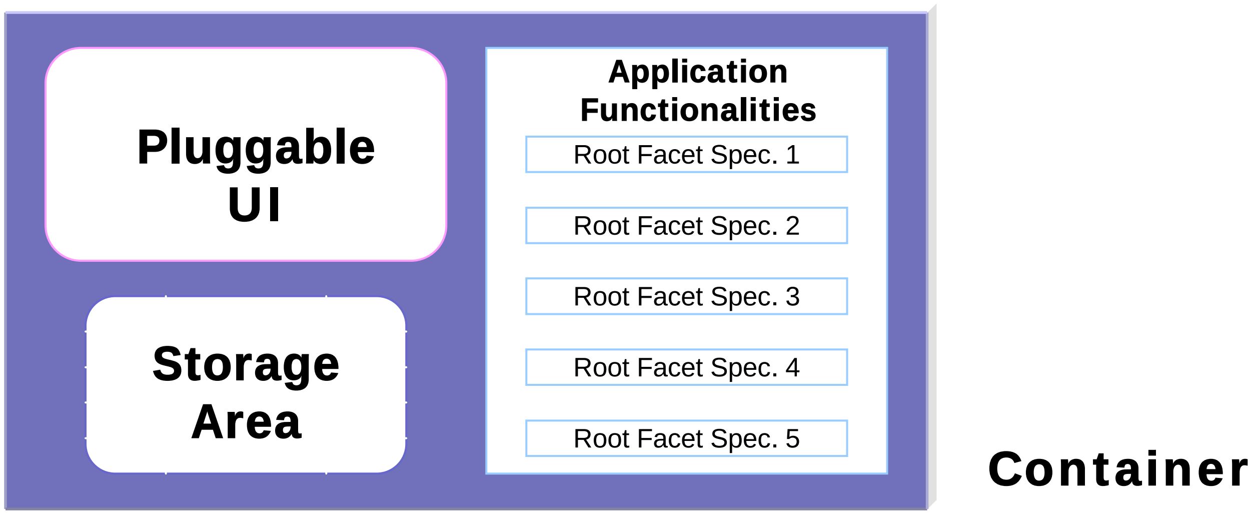

```
....
<dynamic>
    <input_variables>
            <parameter name="m"> 1 </parameter>
            <parameter name="n"> 2 </parameter>
    </input_variables>
    <formula> n^2+5m </formula>
</dynamic>
```

Discarding a Facet

- Every facet is loaded in **its own user class loaders**
- If there are no strong references to any of the classes loaded by the class loader, the class loader can be garbage collected.
- If the class loader is collected, then all the classes that were loaded by the loader will be unloaded from the JVM.

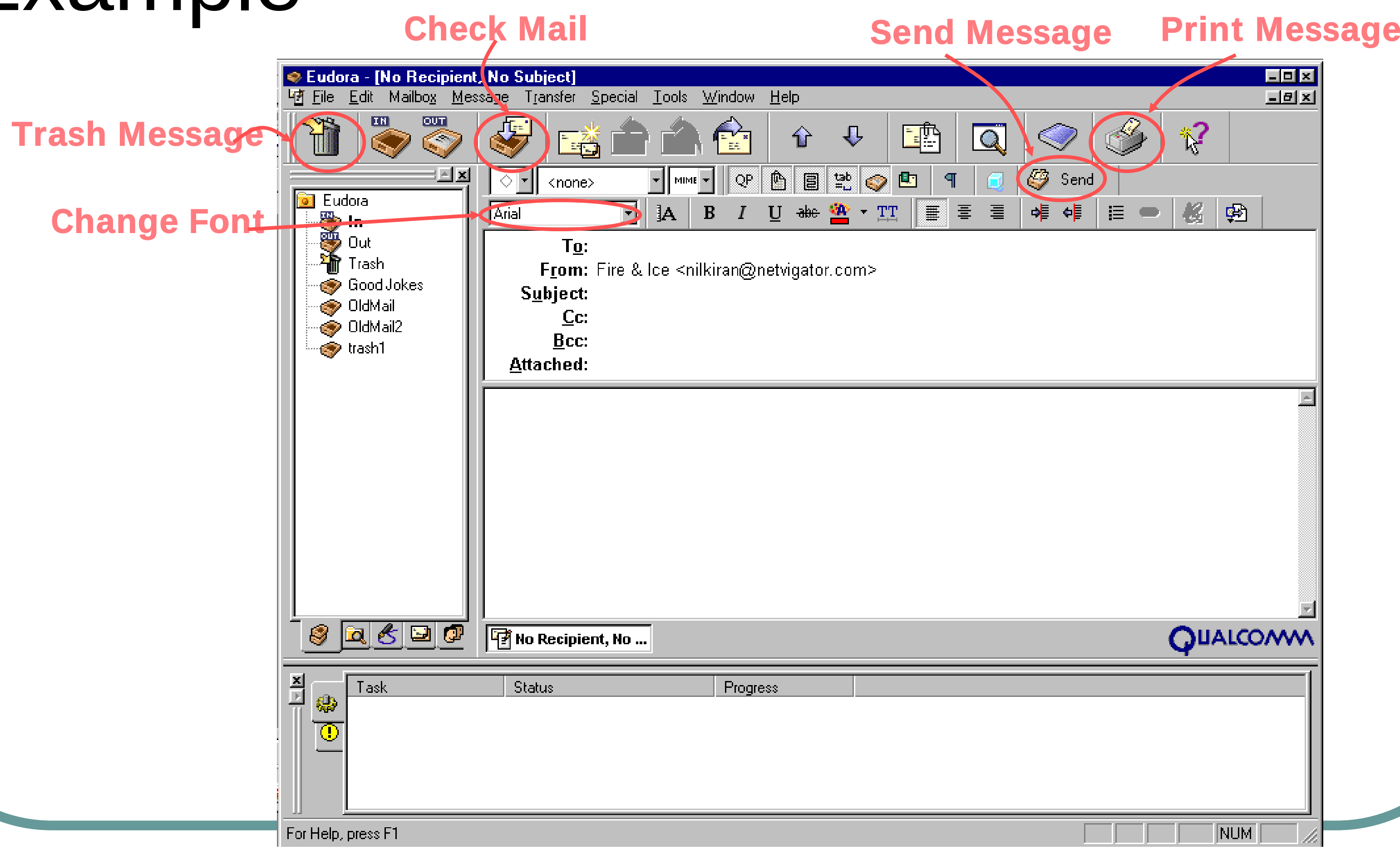
Containers

- ***Application-like abstraction***
  - Interacts with the user through the UI
  - Provides a place to store run-time state
  - Provides Specifications of the **root facets**
- Root facet specification : the functionalities this particular container can offer.



Container and its root facets

Example

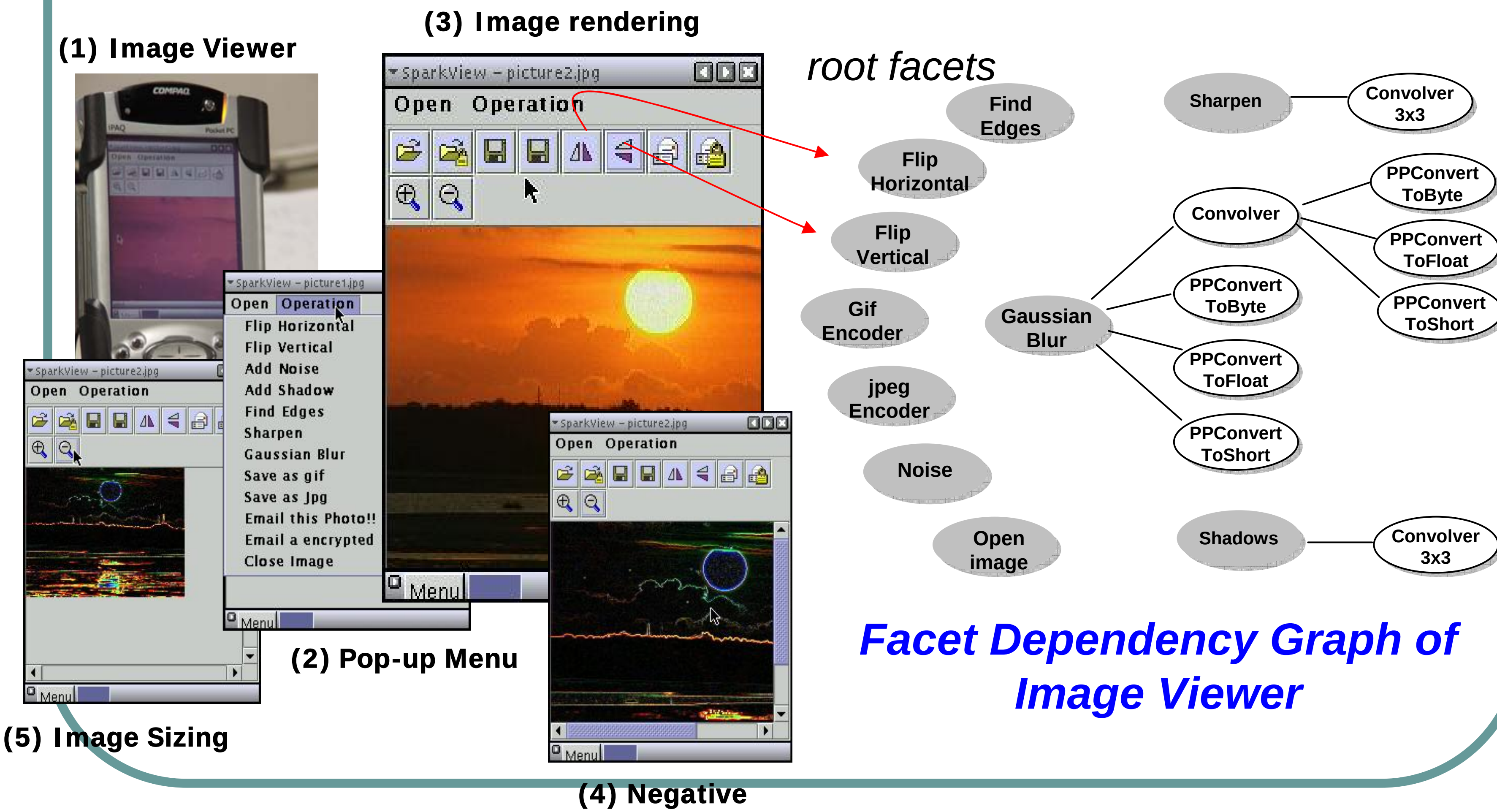


Object-oriented vs. Facet-Based

	Object-Oriented Programming	Facet-Based Programming
Unit of programming	Object	Facet
Granularity	1 class	Can have more than 1 class
Interfaces	Any number of interfaces, with any number of public methods	Only has 1 publicly accessible method, which needs to follow a contract
State and Persistence	Stores some form of state during its lifetime. May contain some persistent state.	Does not store any state between 2 invocations. No persistent state in facets.
Driving Principle	Data-centric	Functionality-centric
Run-time State	Distributed among all instantiated objects	Centralized in container

Sparkle I: Image Viewer

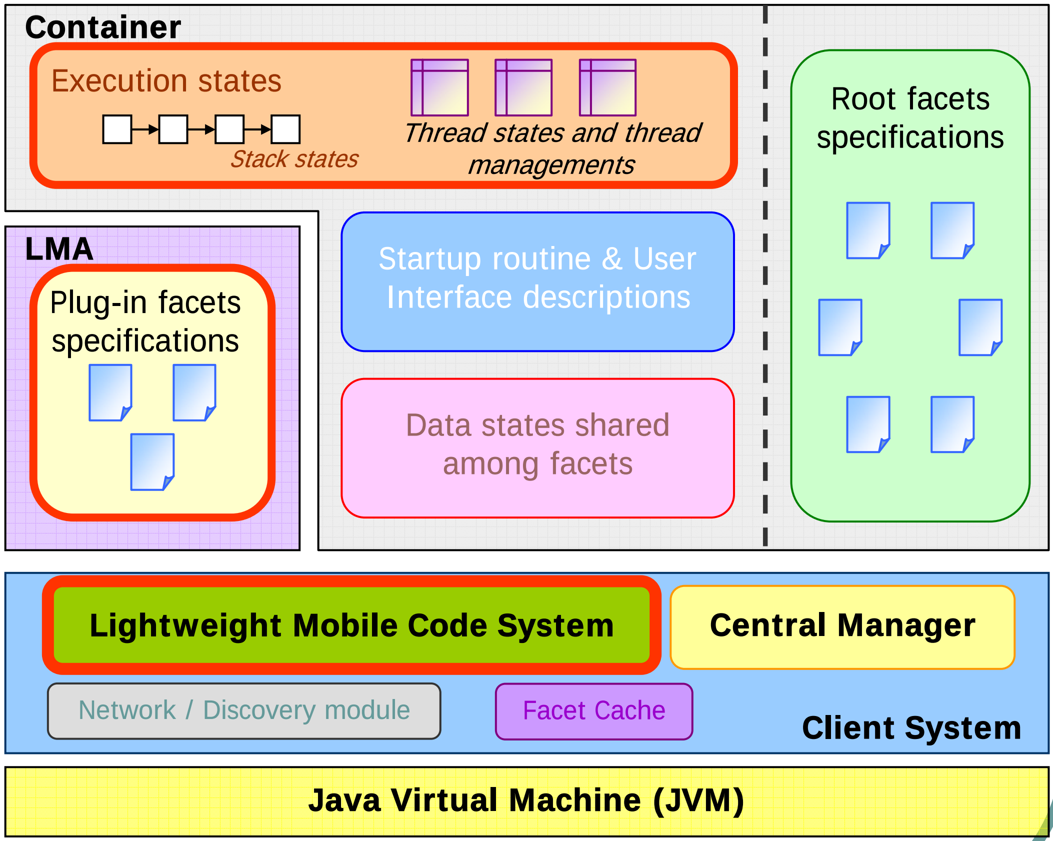
- Developing a real-world application utilizing the facet model



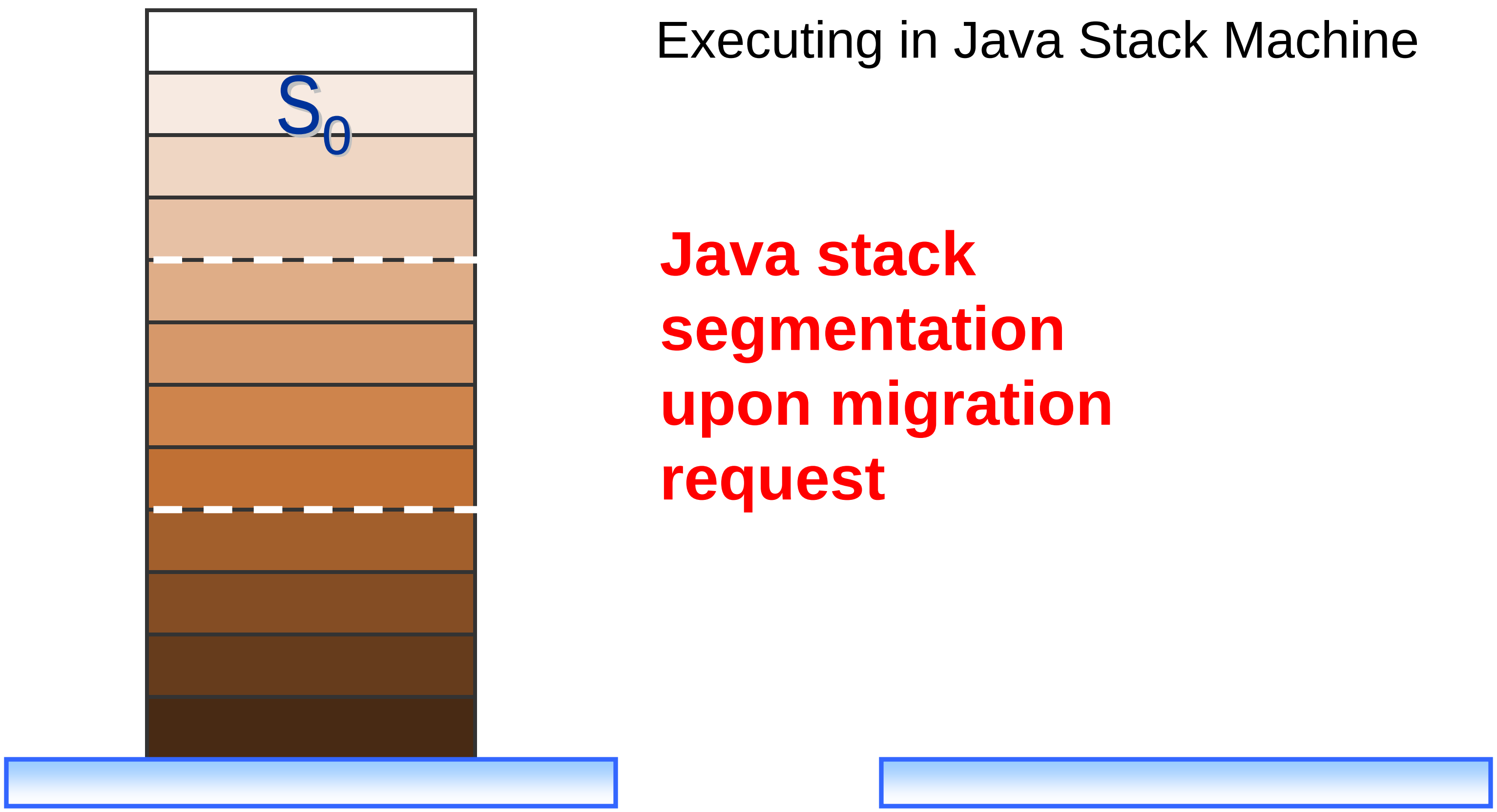
Facet Dependency Graph of Image Viewer

Sparkle I: Strong Mobility Support

- **Core components:**
  - Lightweight Mobile Code System (LMCS)
  - Lightweight Mobile Agents (LMA)
  - Container
- **Uses JavaGo for source code instrumentation and achieve strongmobility. (No modification of JVM)**
- **Incorporate Code-On-Demand (COD) and State-on-Demand (SOD)**



State-On-Demand (SOD)
(Execution Adaptation)



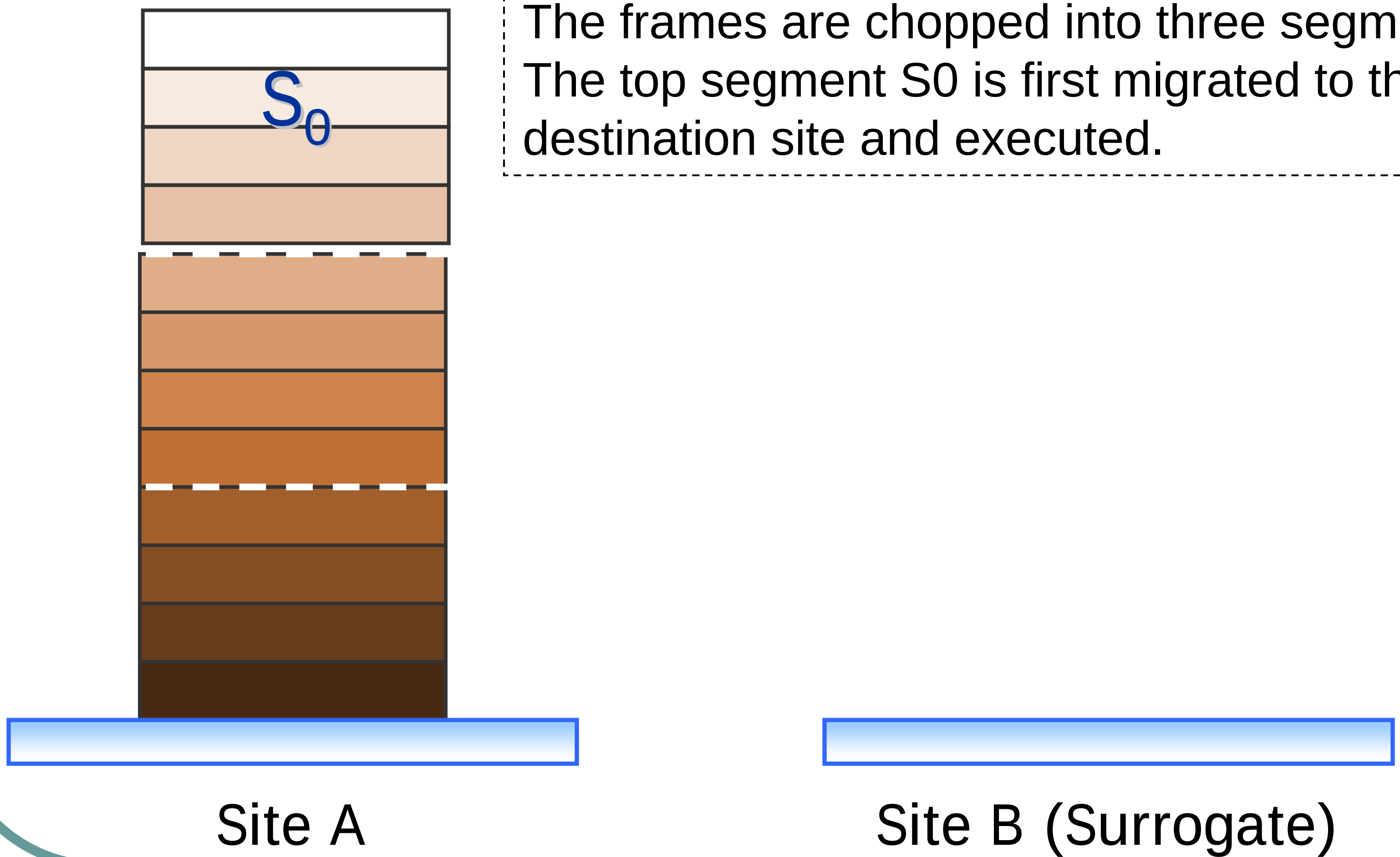
Executing in Java Stack Machine

Java stack segmentation upon migration request

Executing a Java program at Site A

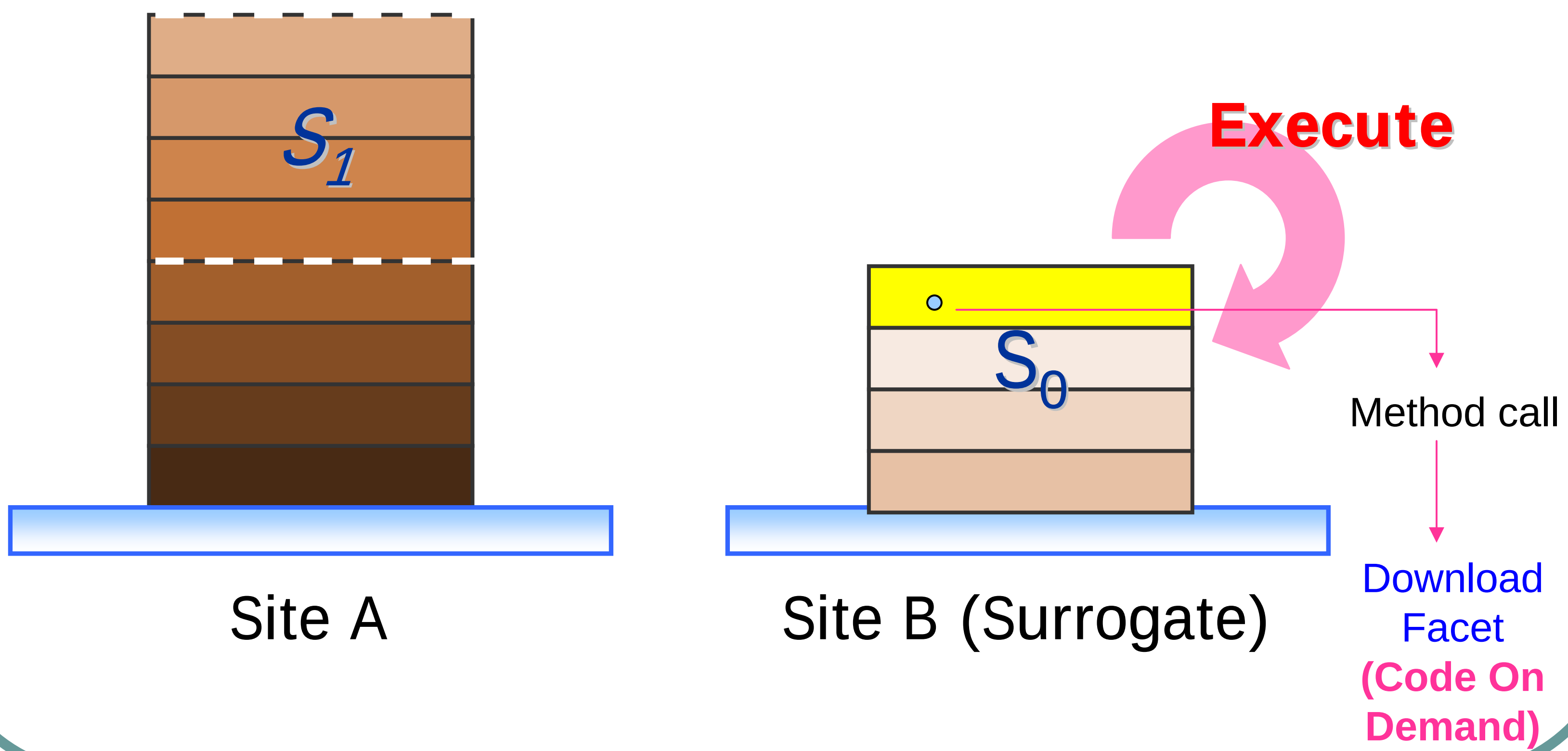
Site B (Surrogate)

SOD execution dynamics

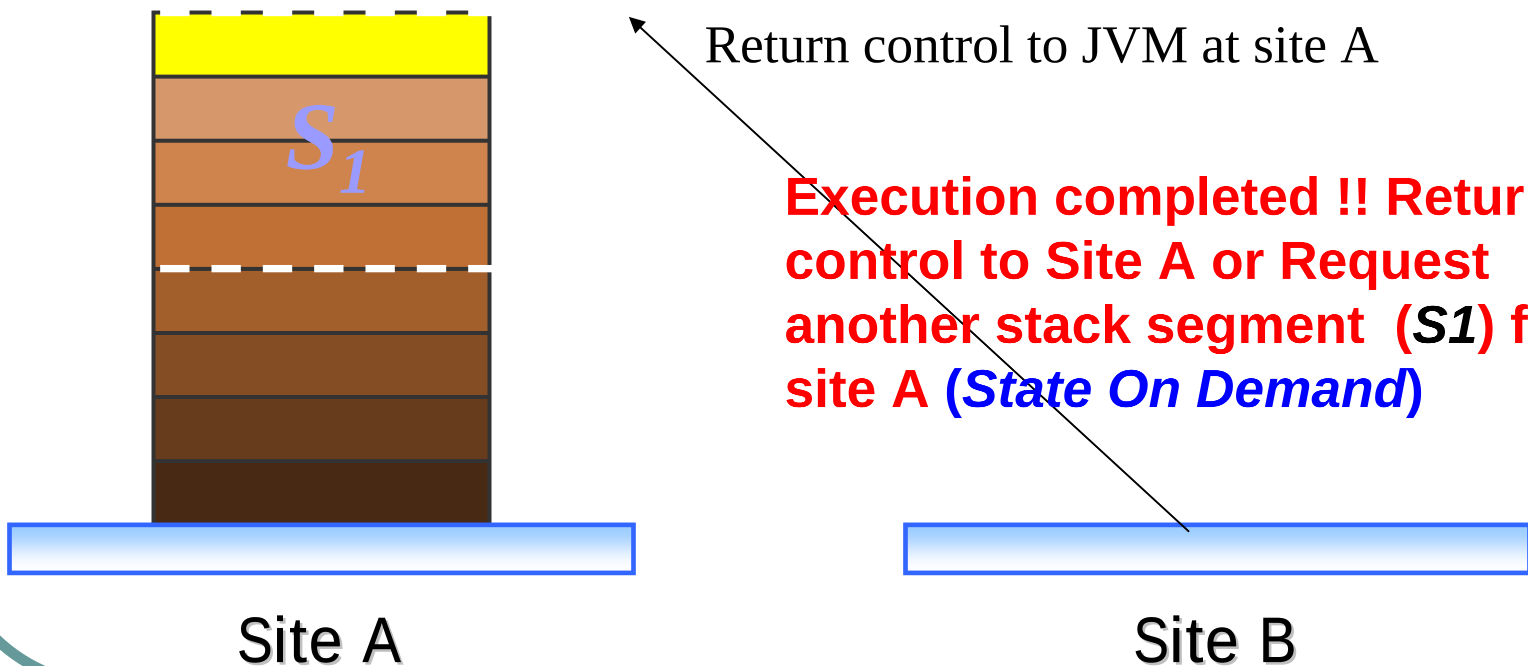


The frames are chopped into three segments. The top segment S0 is first migrated to the destination site and executed.

SOD execution dynamics



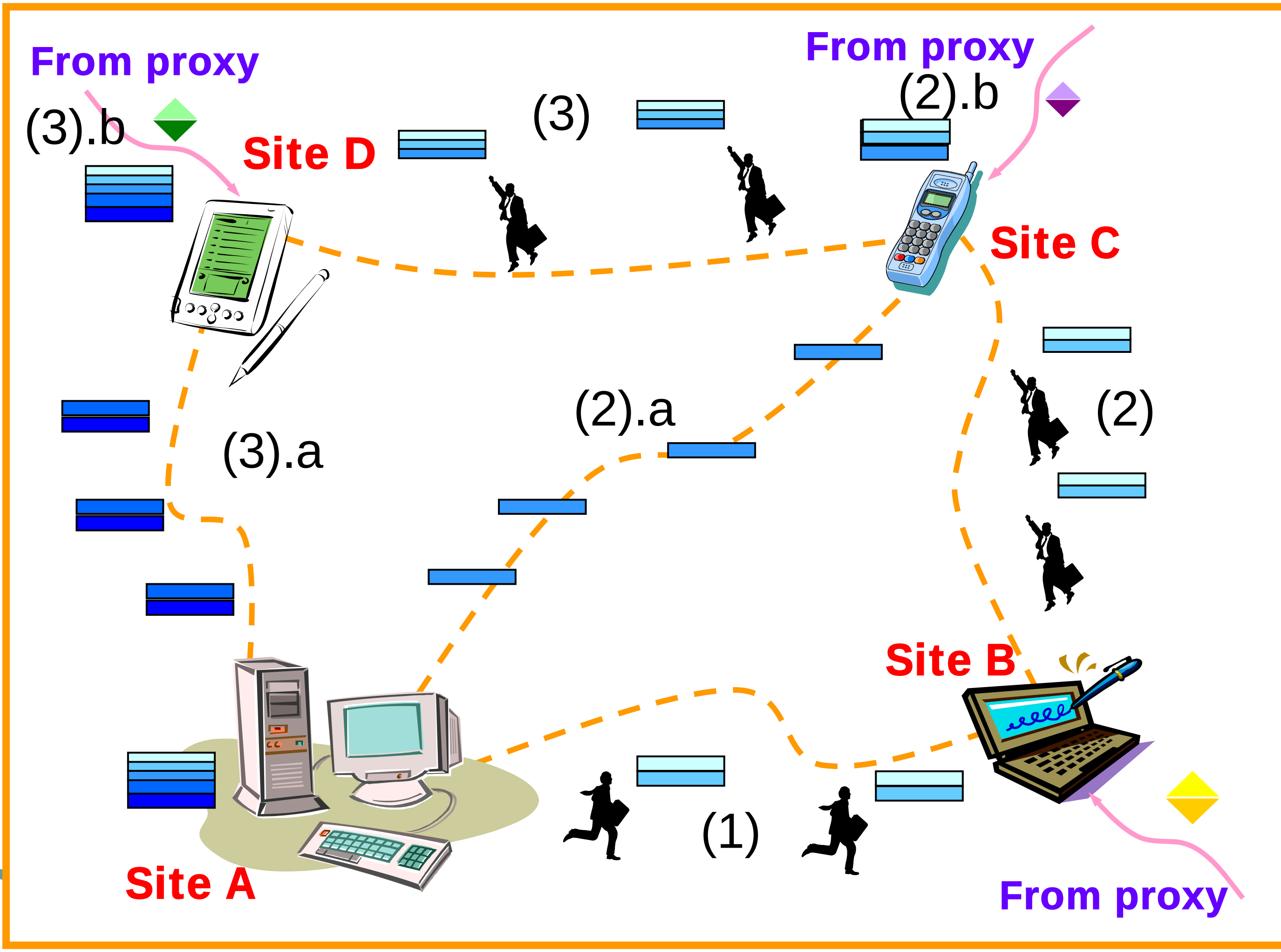
SOD execution dynamics



Execution completed !! Return control to Site A or Request another stack segment (*S1*) from site A (*State On Demand*)

Sparkle I:
Execution Adaptation with SOD + COD

Delay code (Facets) binding after the stack frames migration



SOD Experiment results: Bandwidth Saving

	Fib(35)	Qsort(5000)	NQueen(10)	NQueen-opt(10)
# agent hops	12	12	40	40
(without SOD)	327088	1715062	33600410	33932825
(with SOD)	315197	574655	32183569	16365801
% saved	3.64%	66.5%	4.22%	51.8%

- Fib and NQueen obtained relatively small bandwidth gain (3.64% and 4.22%)
- Qsort and NQueen-opt got high bandwidth gain (66.5% and 51.8%)

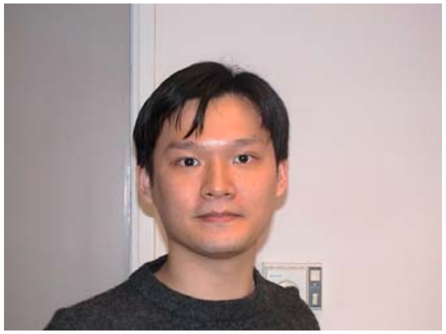
Short Summary on Sparkle I

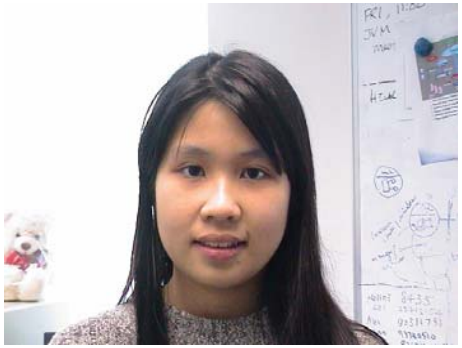
- Main Requirement in Pervasive Computing:
  - Software must be able to adapt dynamically to change and variation
- Functionality Adaptation
  - One of the most versatile adaptation techniques
  - Makes software very dynamic
- Facet Model & Sparkle System
  - Illustrates the feasibility of dynamic component composition in a pervasive environment

Lessons Learned

- **VM Support**
  - Some JVMs lack the required GC support for SPARKLE
- **Connection Speed**
  - Network bandwidth is a problem.
- **Need of a suitable data model**
  - At present, assume it is located locally -> inadequate
  - Need a model defining the location and retrieval of data
- **New software scenario**
  - Anyone can write facets, and people are free to download suitable facet components.
  - Increases competition between software companies and ordinary programmers
- **UI tightly coupled with hardware**
  - Facet concept can be applied to make it flexible

Sparkle I: Theses



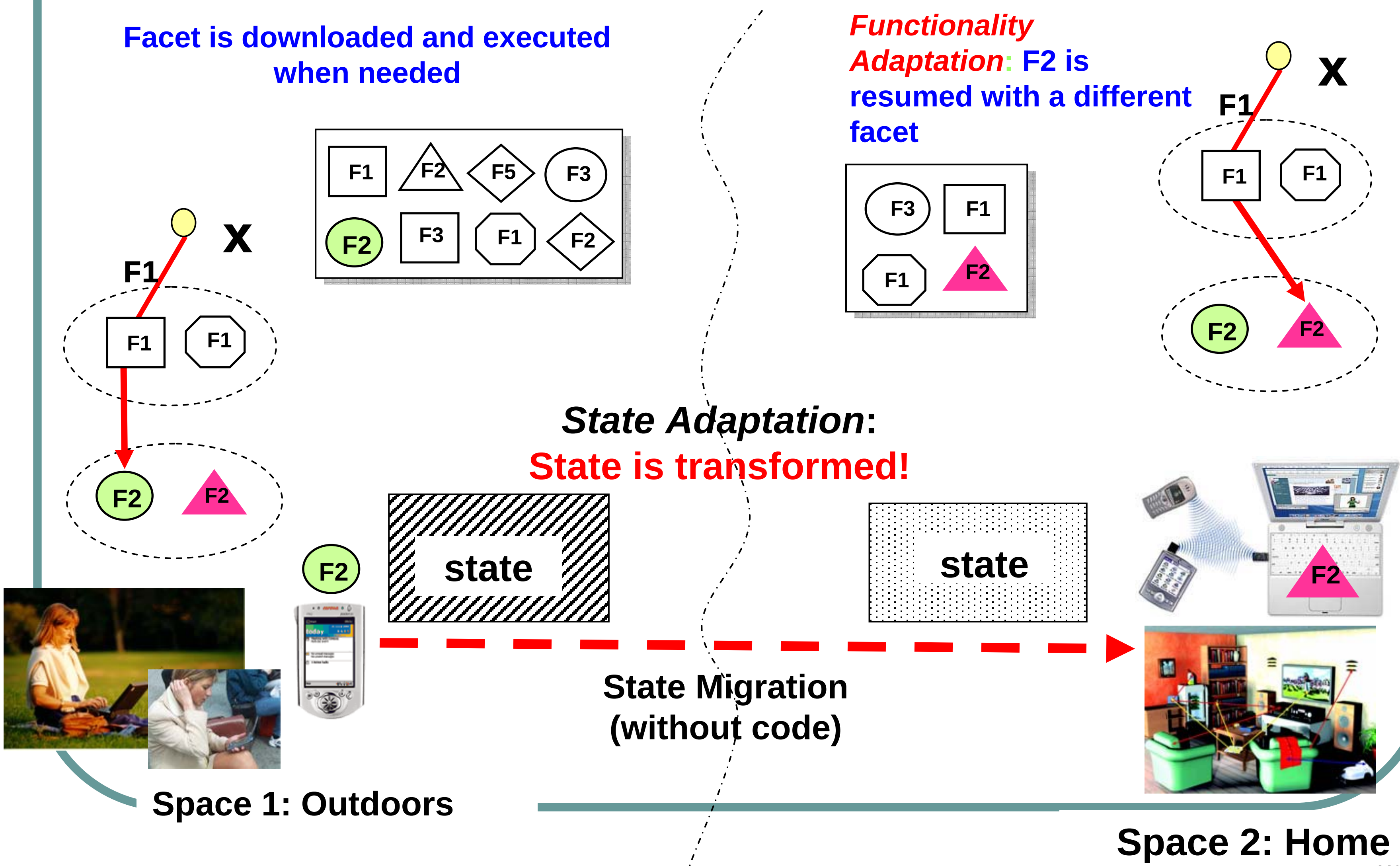


- **Nalini Belaramani (M.Phil, 2000-2002)**
  - Thesis: *A component-based software system with functionality adaptation for mobile computing*
- **Yuk Chow, (M.Phil, 2000-2002)**
  - Thesis: *A Lightweight Mobile Code System for Pervasive Computing*
- **Vivien Kwan (M.Phil, 2000-2002)**
  - Thesis: *An Intelligent Proxy Server System for Pervasive Computing*

Sparkle II : Semantic Adaptation

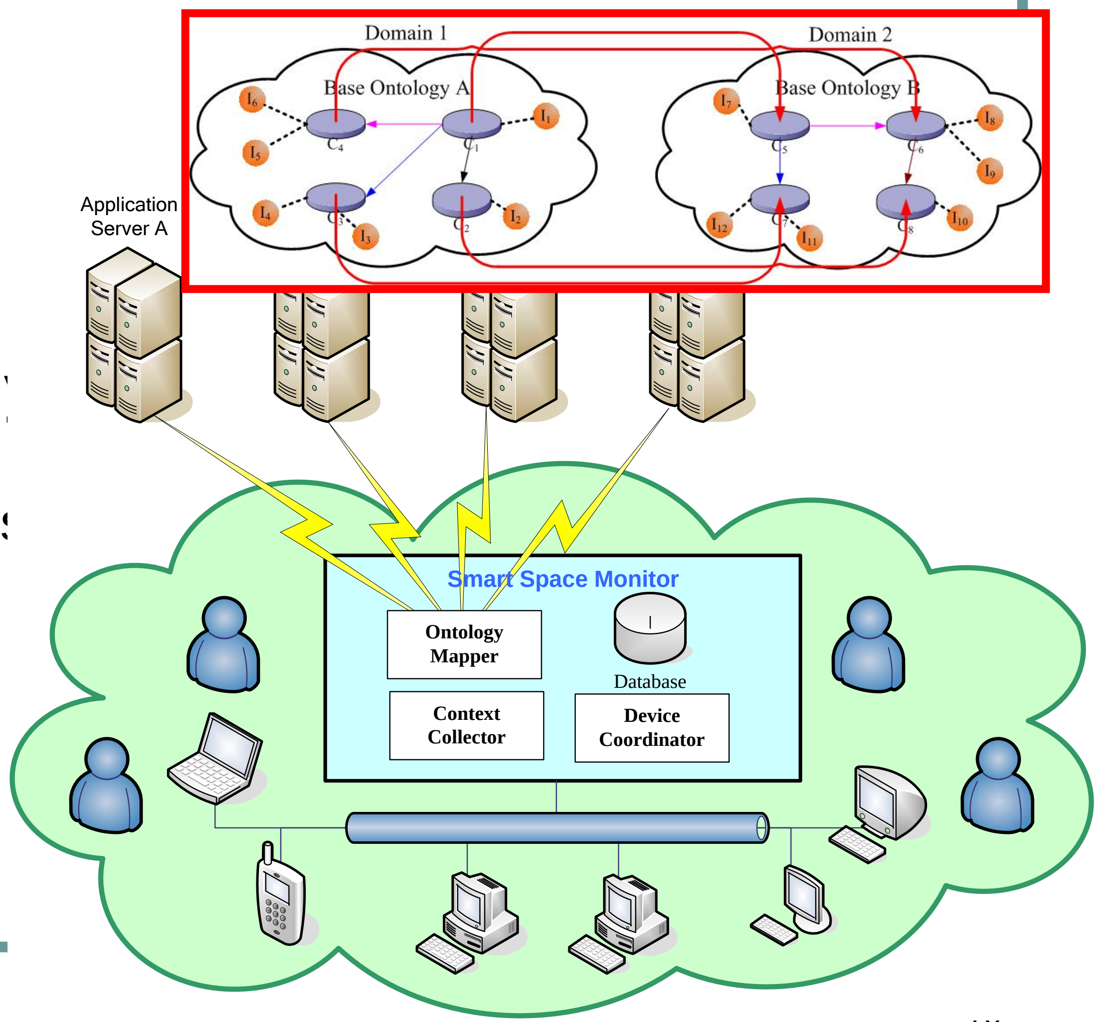
- Context-aware State Management
  - To migrate from one environment to another environment meeting the context changes flexibly and efficiently.
  - E.g., music playing move from office to meeting room
- Ontology-based Knowledge Mapping
  - for basic context awareness

Sparkle II :
Context-aware State Management



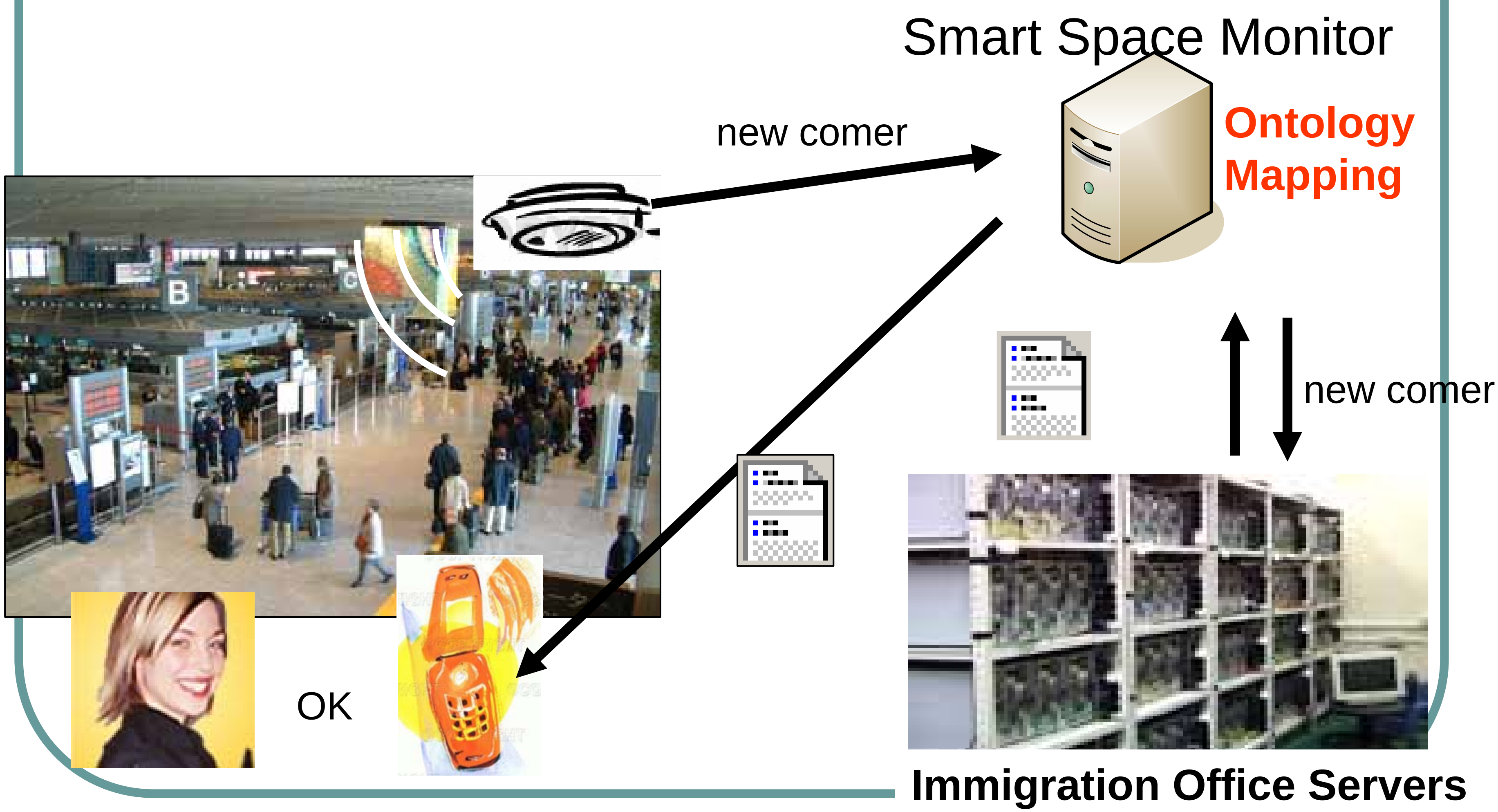
Ontology Mapping

- **Domain ontology**
  - Smart space context, resources, activities done.
  - One in each smart space
- **Application ontology**
  - Device configuration, application parameters, service descriptions
- **User ontology**
  - User identify, social status, user preferences



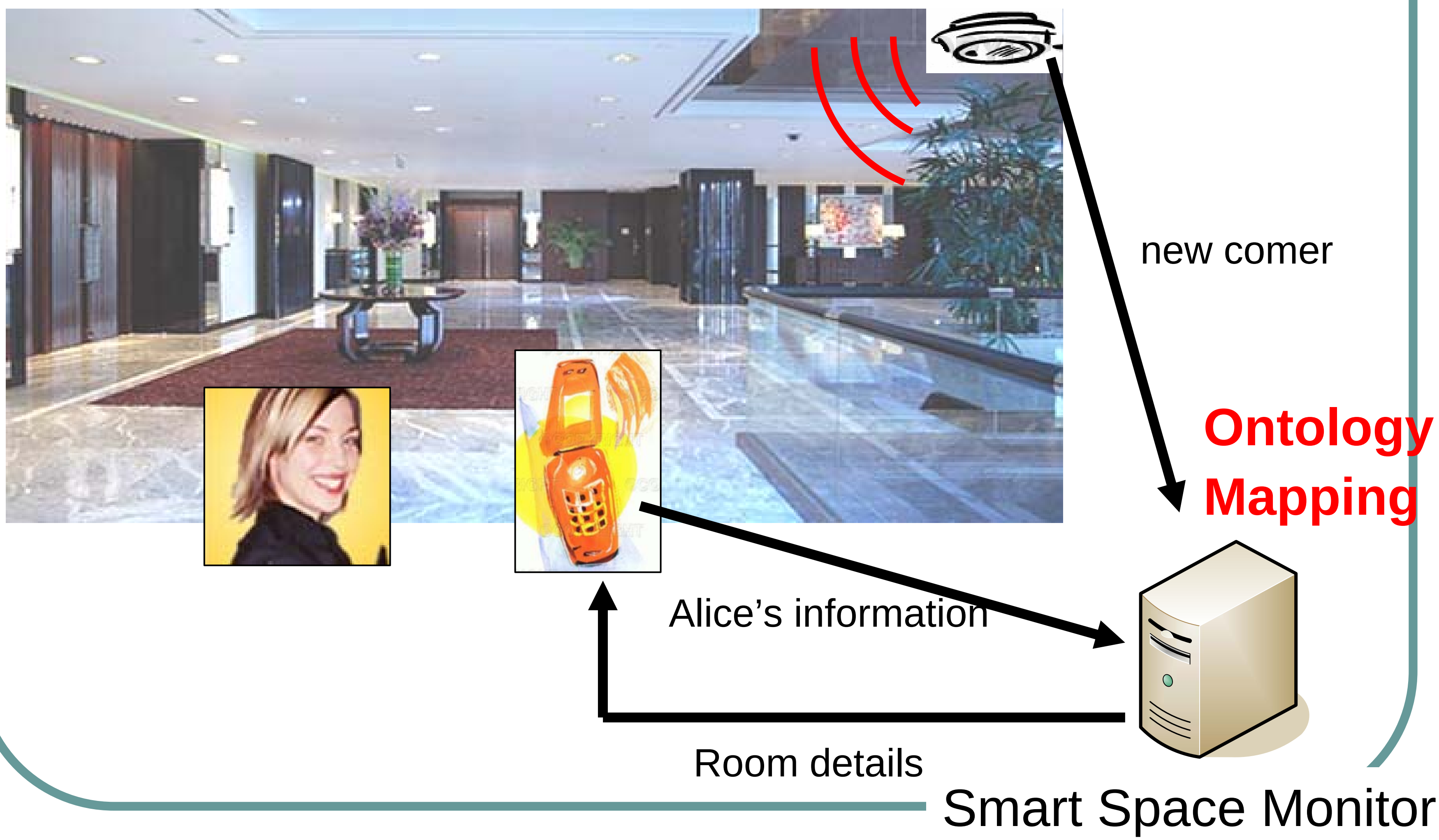
Ontology Mapping

Scenario 1 (Airport Custom)



Ontology Mapping

Scenario 2 (Hotel Check-in)

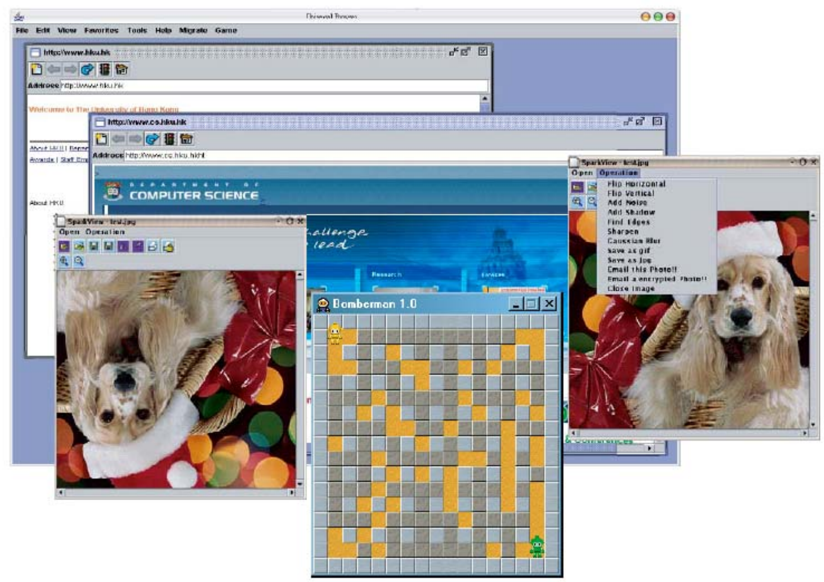


Ontology Mapping : Evaluation

- **Average 82.5% accuracy**
- **Accuracy**
  - Twice more than source-based
  - 4% more than instance-based
- **Efficiency**
  - Much slower than source-based
  - 50% faster than instance-based
- **Space**
  - Runtime memory usage depends on the size of source ontologies and JVM setting
  - The maximum memory usage in our experiments is 300M bytes
- **Limitation**
  - Ontology parsing is time-consuming and huge memory consumption
  - Only Jena parser supports most features proposed by OWL

Sparkle II : Universal Browser (UB)

- The UB targets “browsing whatever you want”. The special graphical user interface allows users to dynamically retrieve the functionalities they want, such as playing games, editing photos etc.



Evaluation

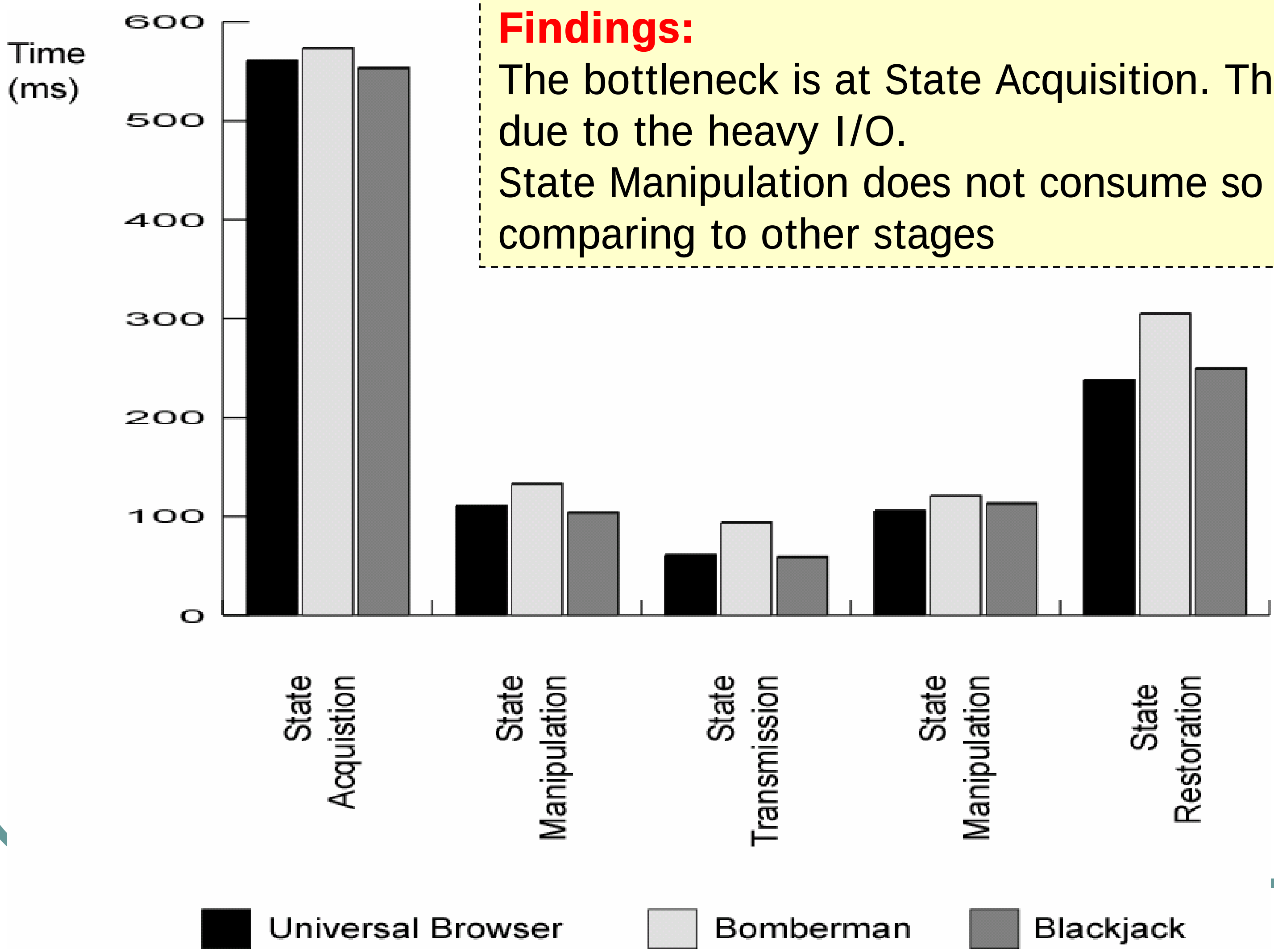
Comparison of latency and data transferred

Applications	Migration Latency
Universal Browser	3837 ms
Bomberman	4038 ms
Blackjack	3933 ms

Findings: The amount of data transferred is reduced although the result is not so significant. Migration time is acceptable in a WLAN.

Evaluation (cont'd)

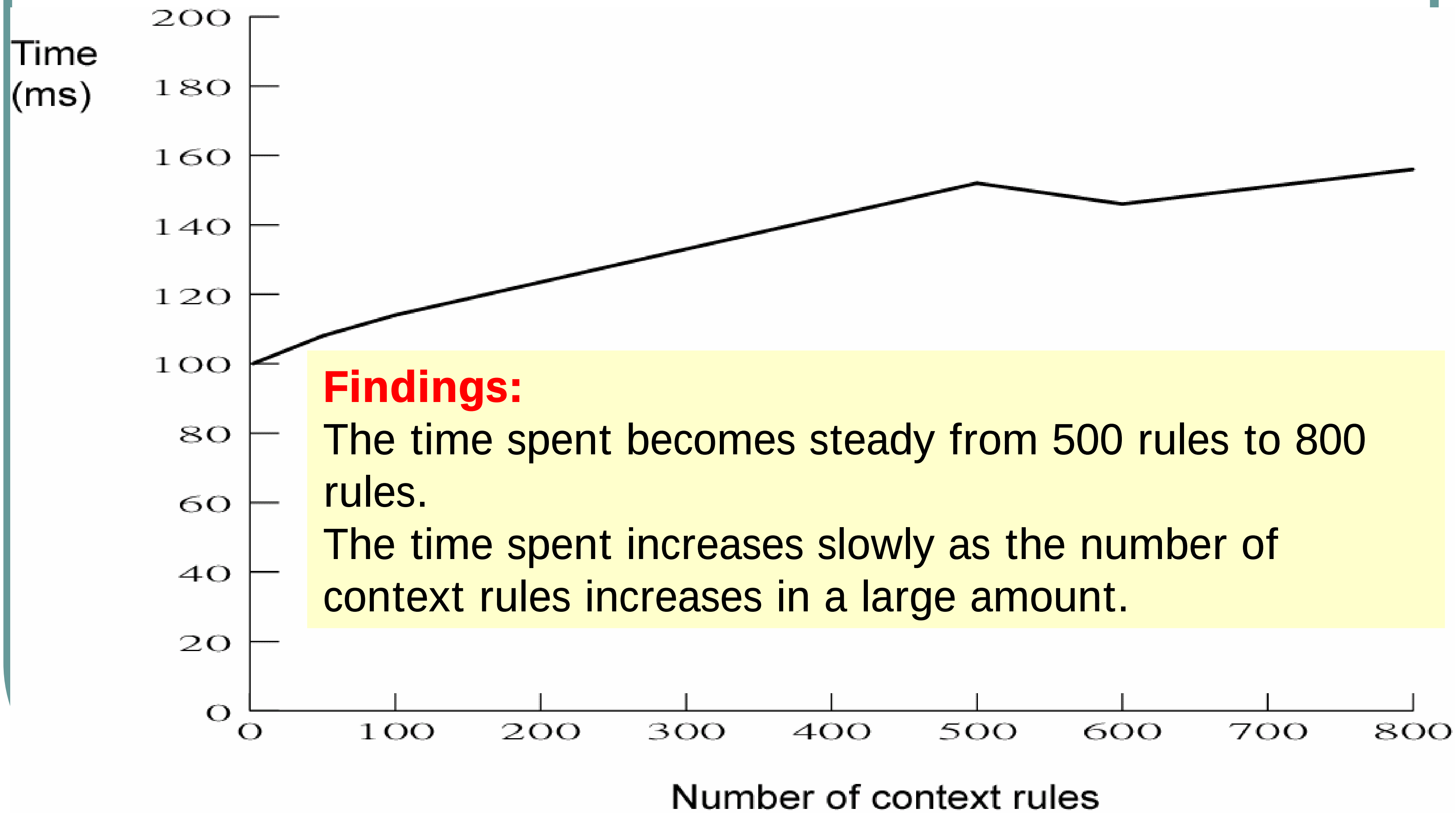
Time Spent for each migration stage



Findings:
The bottleneck is at State Acquisition. This is mainly due to the heavy I/O.
State Manipulation does not consume so much time comparing to other stages

Evaluation (cont'd)

Time Spent against No. of Context Rules



Findings:

The time spent becomes steady from 500 rules to 800 rules.

The time spent increases slowly as the number of context rules increases in a large amount.

Sparkle II: Contributors

- **Siu Po Lam (M.Phil, 2002-2004) :**
  - **Thesis: *Context-aware State Management for Pervasive Computing***
- **Kong Choi Yu (M.Phil, 2002-2004) :**
  - **Thesis: *Effective Partial Ontology Mapping in a Pervasive Computing Environment***

Sparkle III : Smart Instant Messenger

Sparkle III: Smart Instant Messenger

- Pervasive Communication
  - Anytime, anywhere
  - "Anything"
  - In a *buddy-like* way
    - Appropriate
      - **Knowing when, where, how**
    - Familiar
      - "**gd nite & cu tmr**"
      - Use your own dialect
- *This project looks at the potential usage of IM on mobile devices in future pervasive environments.*

P – Person

D – Device

S – Software

O – Other entities

	P	D	S	O
P	✓	✓	✓	✓
D	✓	✓	✓	✓
S	✓	✓	✓	✓
O	✓	✓	✓	✓

Pushing IM into PCE

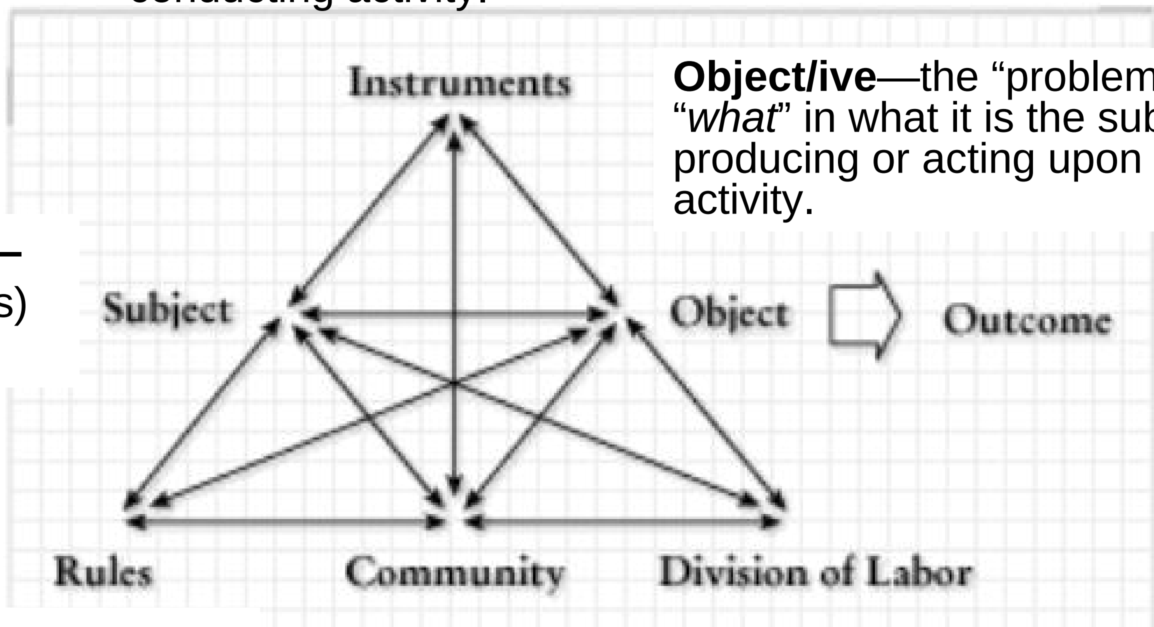
- Everything as your buddy and can be communicated using real-time message exchange
- Three main features
  - Context-aware presence management
    - Context as presence
    - Different buddies see different status
  - Resource buddy services
    - extend the concept of "buddies" to all software and hardware components in your working space
    - IM as the unified communication interface
    - Buddy understands your dialect
  - Dynamic grouping
    - Location-based Grouping ("buddy discovery")
    - Activity-based Grouping ("task centric")

Context Modeling: Activity Theory

Instruments (Tools)—cultural artifacts (material, symbolic) that provide the "means" for conducting activity.

Object/ive—the "problem space," or the "*what*" in what it is the subjects are producing or acting upon through their activity.

Subjects— individual(s) in activity



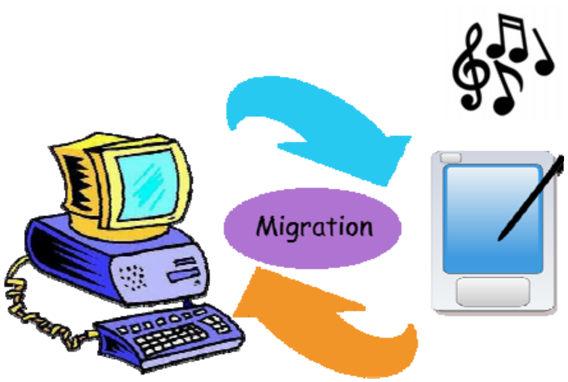
Rules—principles, expectations, norms, and conventions that constrain or regulate how an activity is carried out

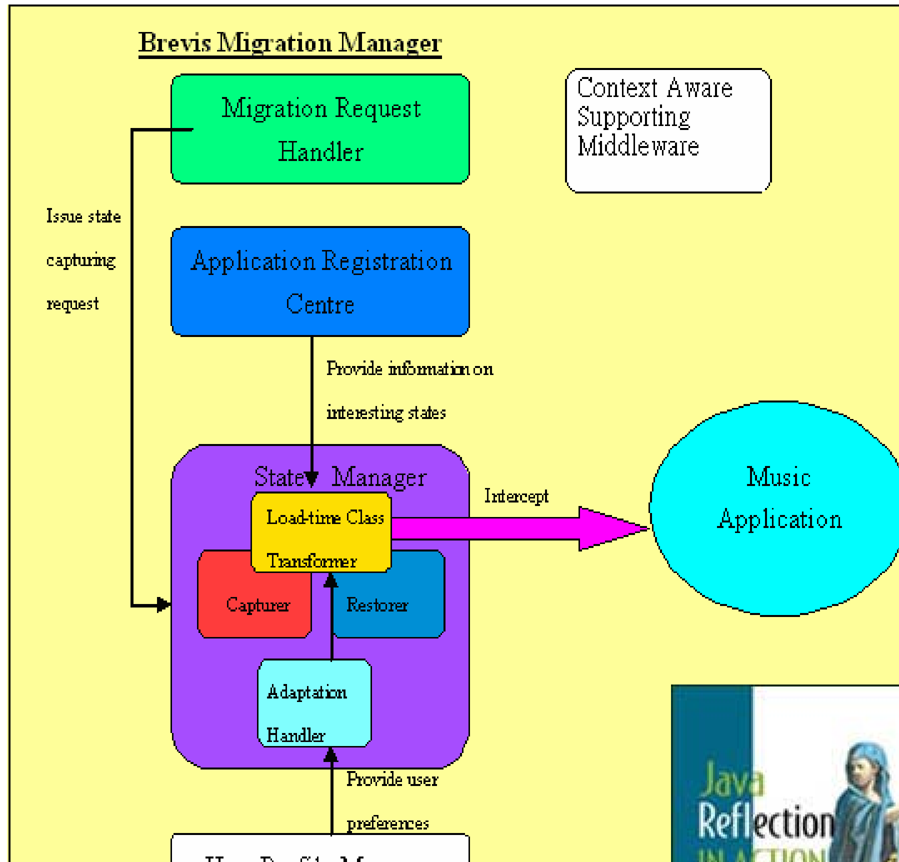
Community—people who share the same object/ive

Division of labor—allocation of tasks and responsibilities

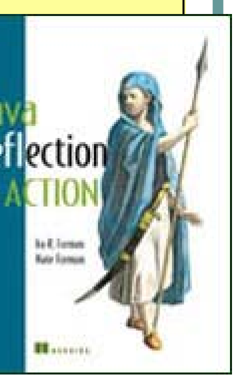
Mobility Support in Sparkle

- Java Media Framework
  - Java package to implement media player
- Reflection-based state capture and restore mechanism
- Context-aware state adaptation



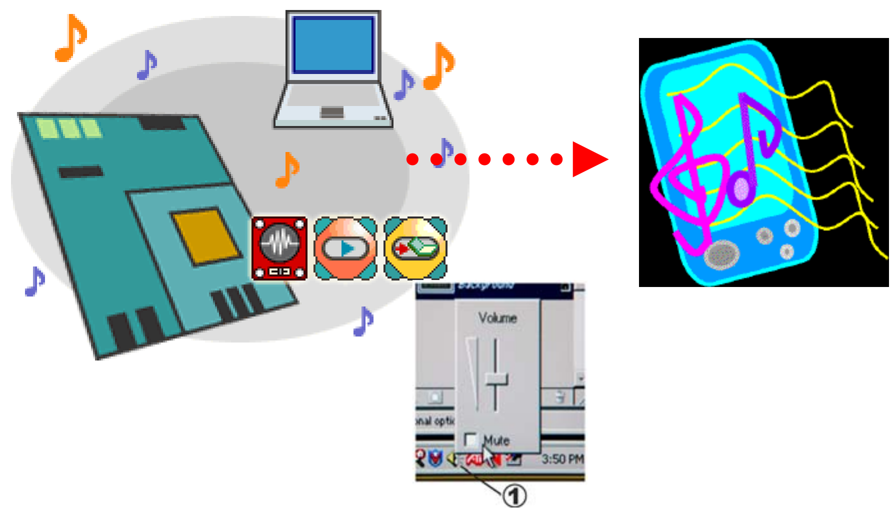


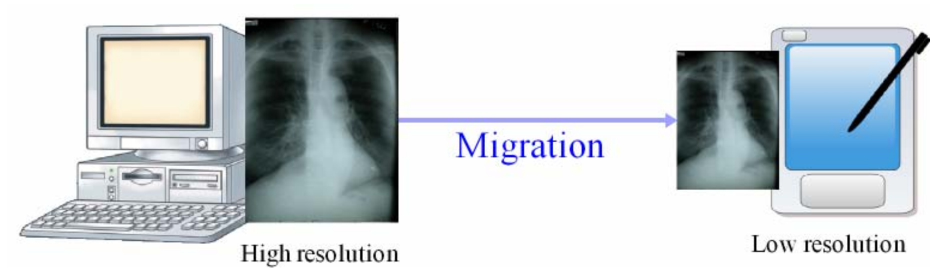
Brevis Migration Manager



Brevis Migration Manager

- ***State adaptation***
  - Change the states captured before restore
    - Volume of music playing based on activity
- ***Data adaptation***
  - Change the data used by application
    - Data format, size and resolution, data availability
- ***Cross machine and platform adaptation***
  - Migration can across different devices
    - PC to PDA, PDA to PC



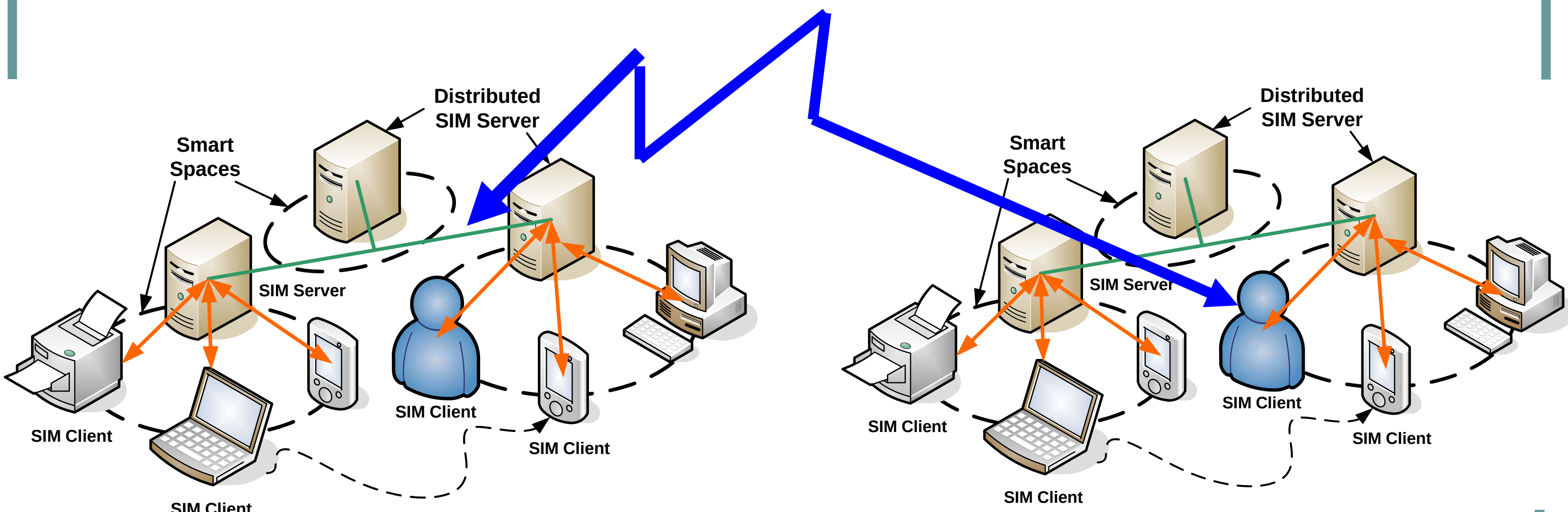


States capture and restore in Brevis

- By Java Reflection technology
  - Reflect states in dynamically loaded class
    - Retrieve the state information by reflecting IM
    - Save and transmit the states
      - States could be stored in fields
    - Receive the states and injected into the running IM program
- States captured and restore
  - User account information
  - Chatting information

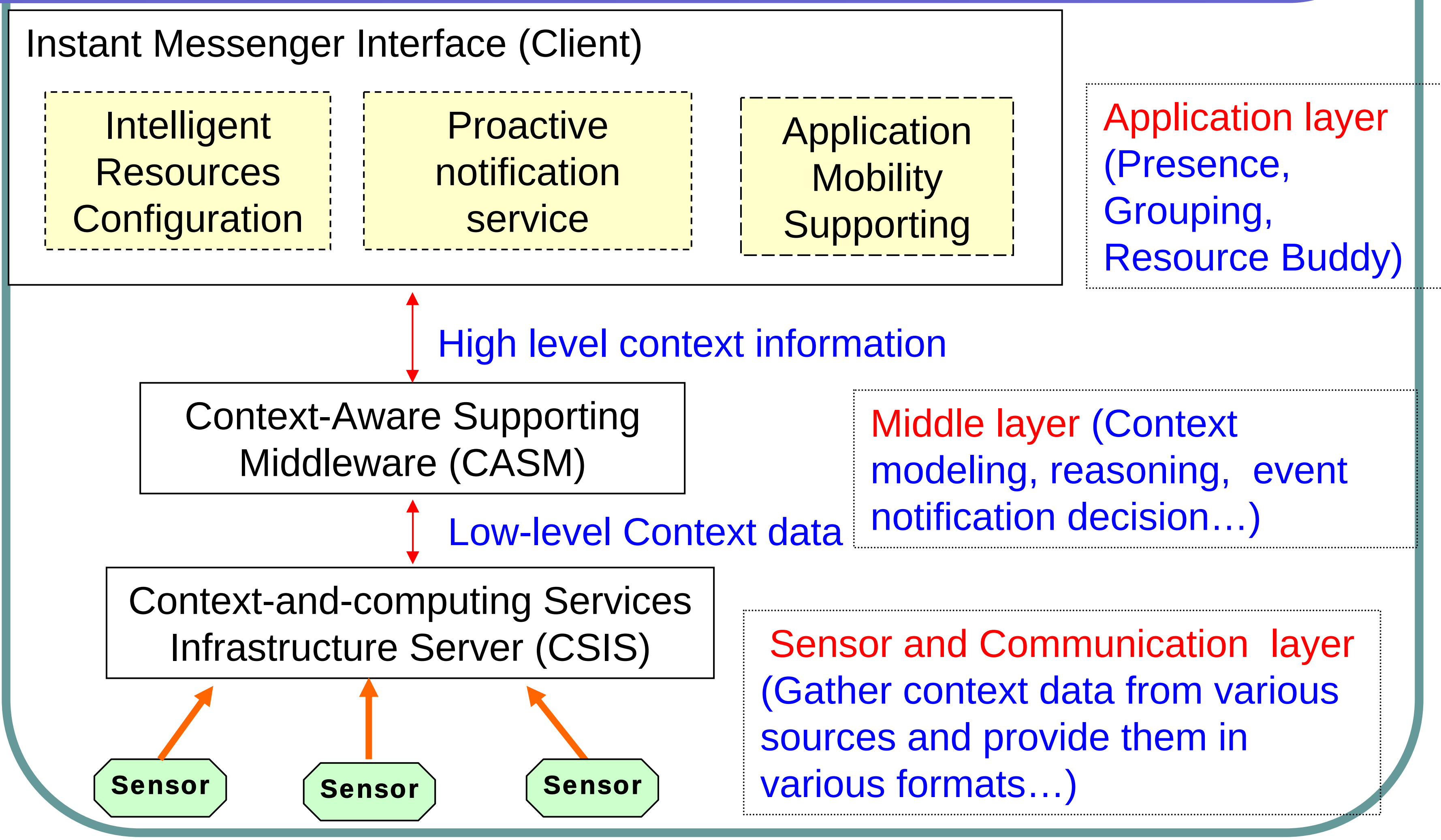
Deployment of SIM

- **Extend the IM framework and implant context-aware behaviors**
- **Separate context provision from context consumption**
- **Everything′s behind an SIM client**
- **Distributed Servers Architecture**

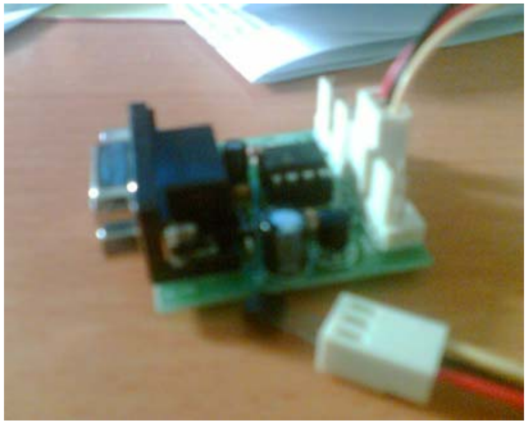


(work space 1) (work space 2)

Internal Design of SIM

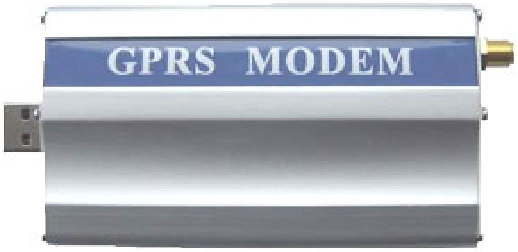


Hardware of SIM



temperature logger

Location Tracker
RFID Tag and Reader



GSM/GPRS Modem

Bluetooh

Speaker as Noise
Detector

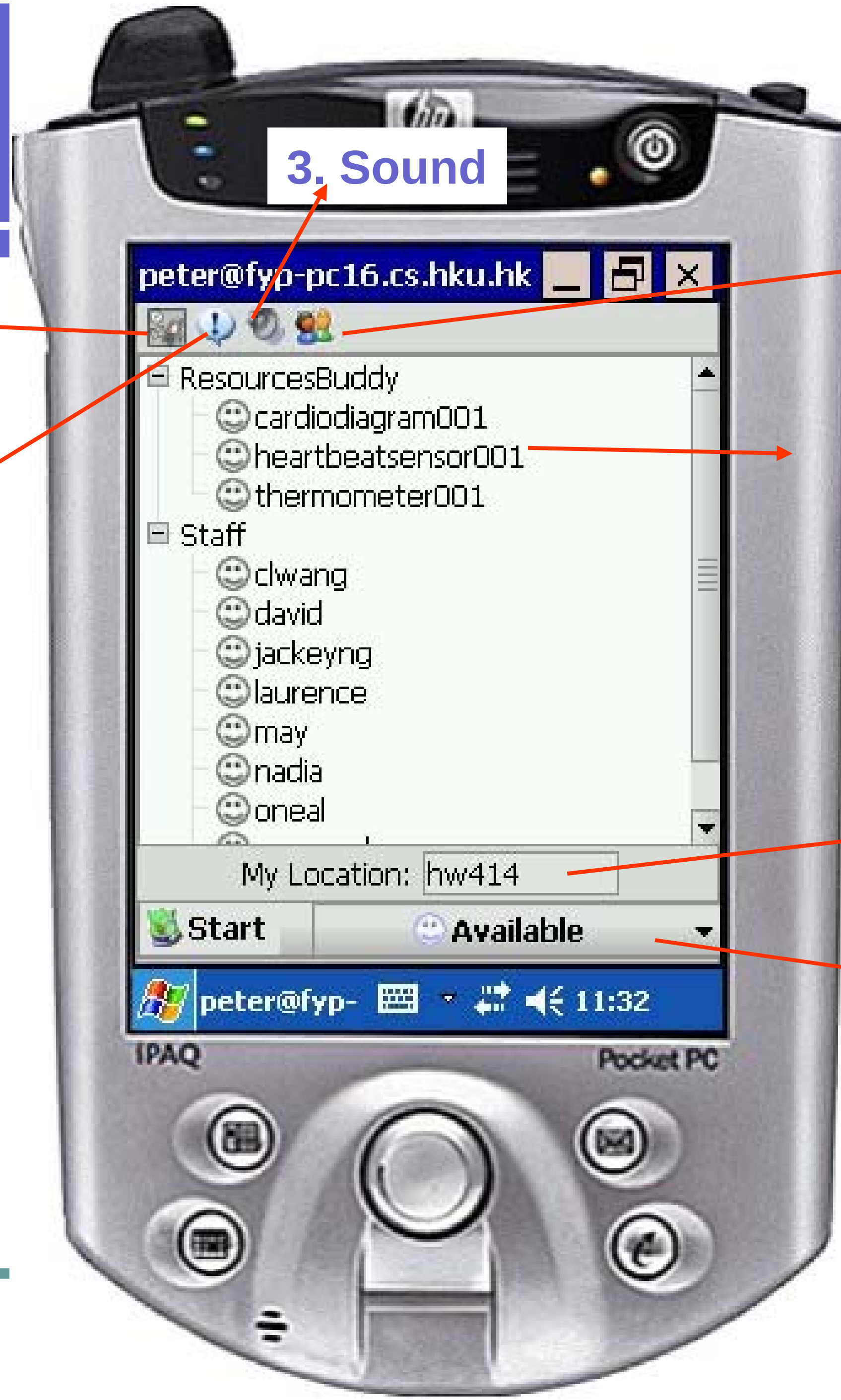
WebCAM as Motion/Light
Detector

Performance & Screenshots

Main Panel

1. Reminder

2. Unread system message



4. Show Offline buddies

Resource buddy

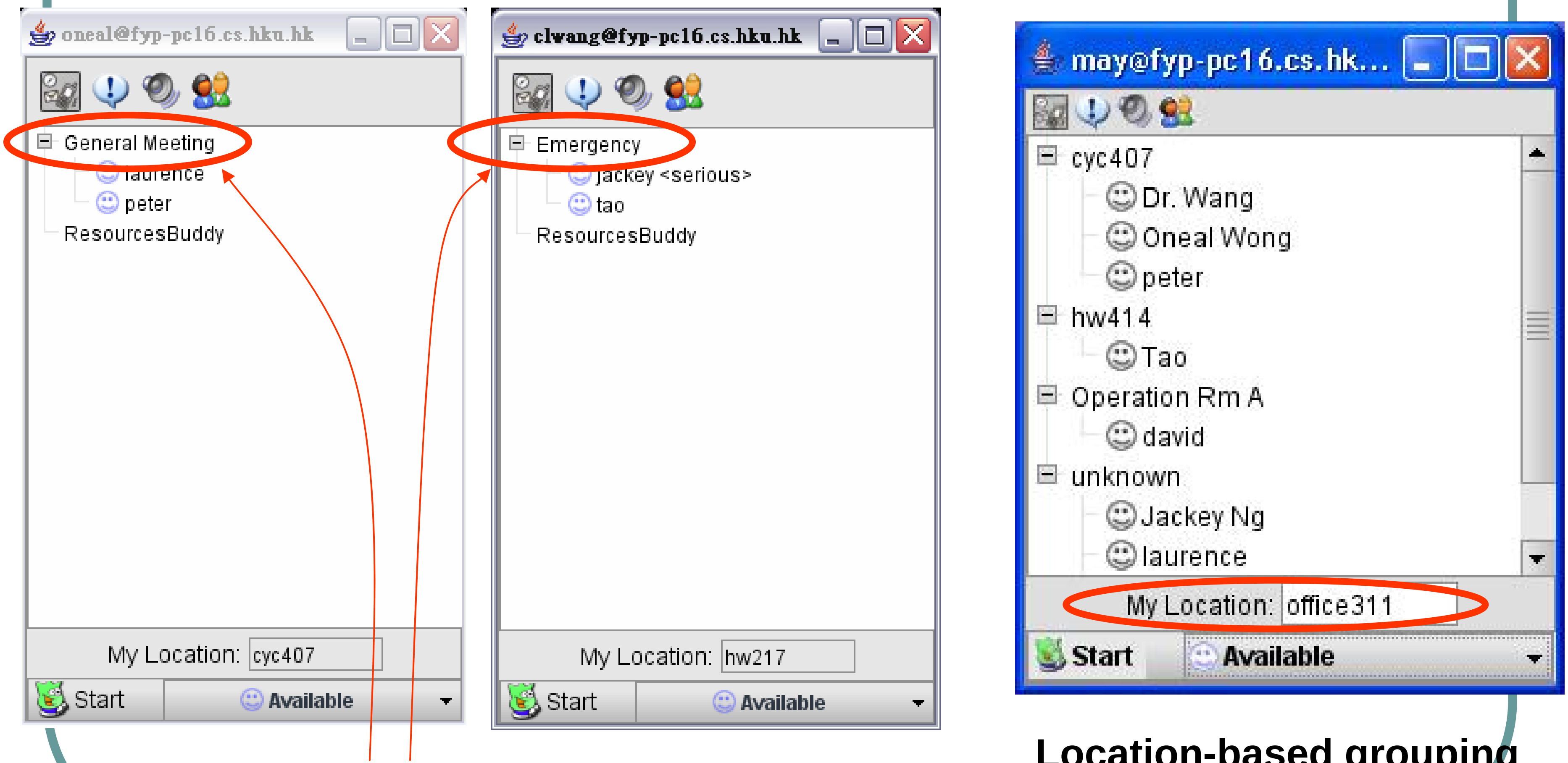
Location

Presence

IM Feature - Reminder

- Criteria:
  - Time
  - Location of target buddy
- Forms of message:
  - Text Form
  - Voice Form
    - rely on Text To Speech Technique

Dynamic Grouping

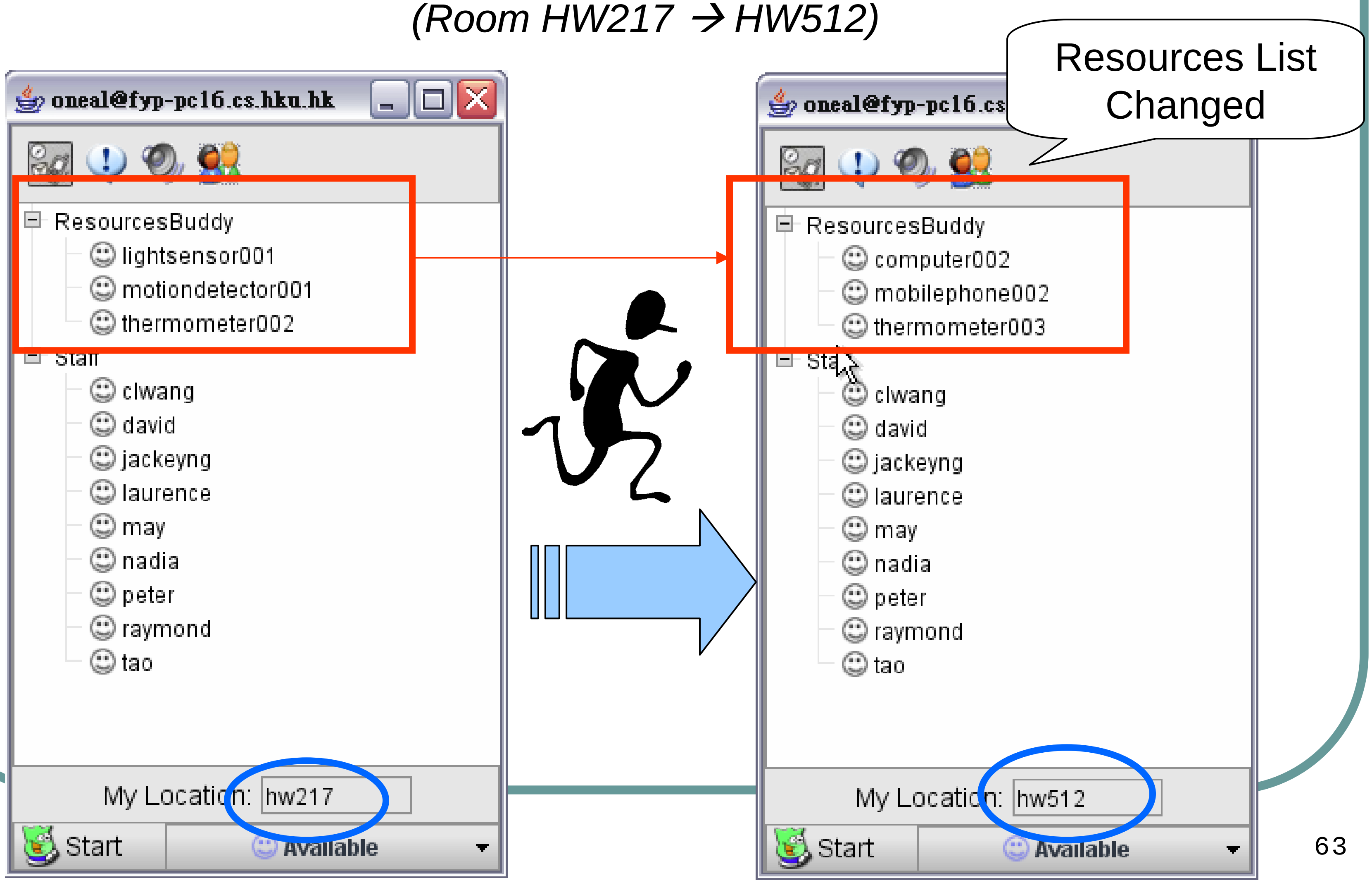


Location-based grouping

Activity-based grouping

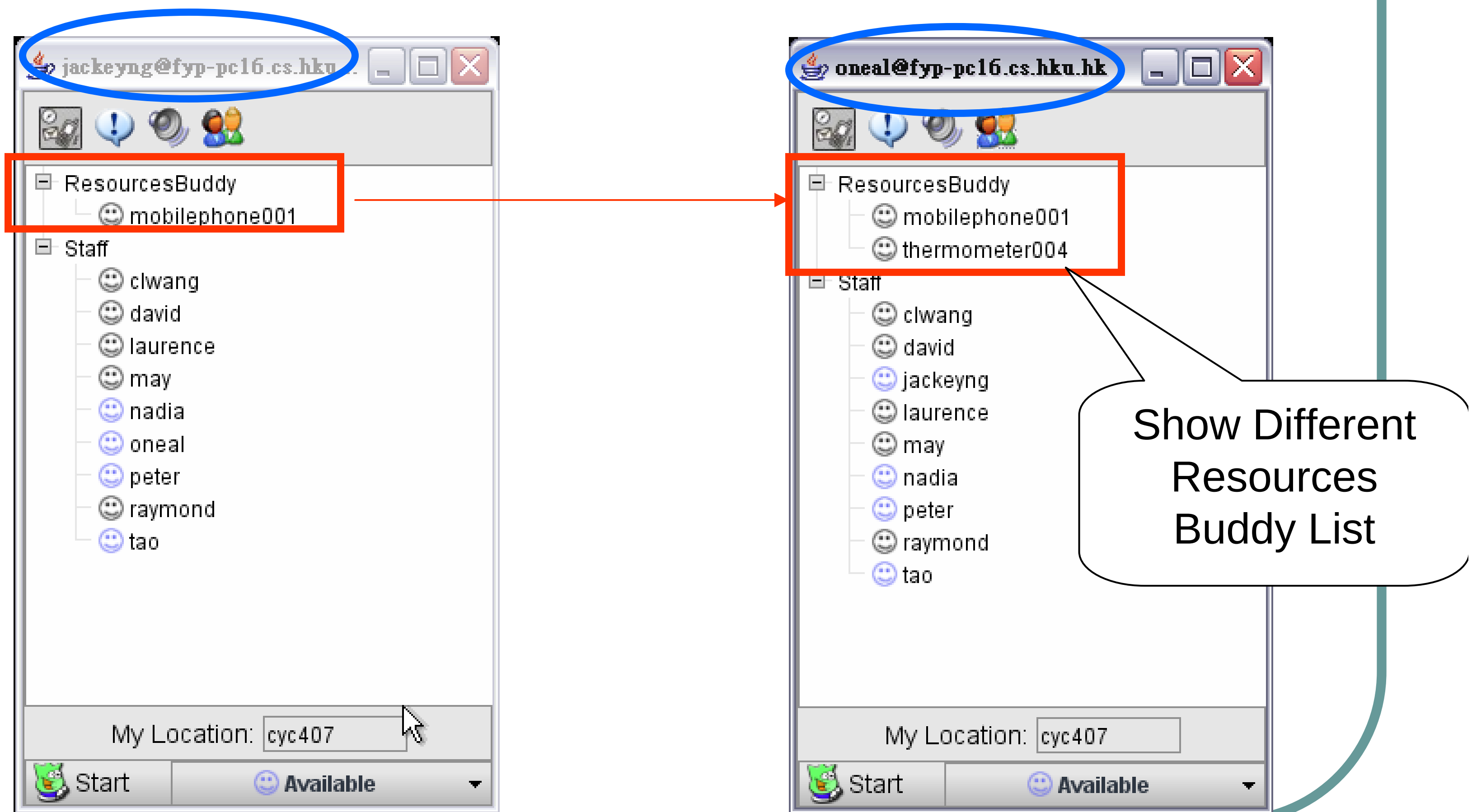
Adaptive display

Same person at different locations
(Room HW217 → HW512)



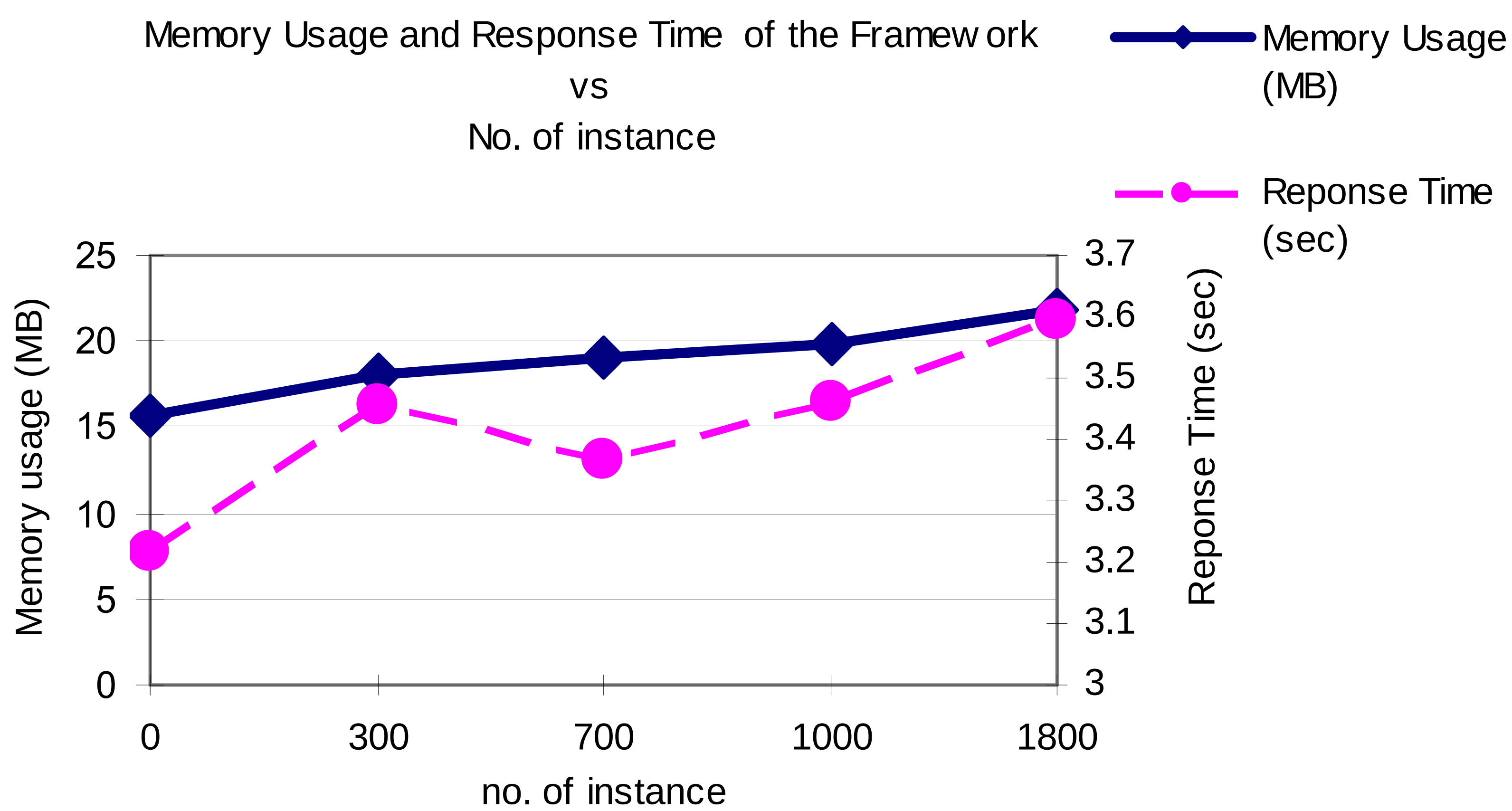
Adaptive display

Same location with different persons



(Screen display when Jacky and Oneal enter Room CYC407)

Performance Evaluation



Sparkle III: The SIM Team

- **Research Students**
  - Ms. Xiaolei Zhang (Ph.D)
  - Mr. Hauyu Huo (Ph.D)
- **2004-2005 FYP students**
  - Law Chun Fai (Terry)
  - Chan Sung Ming
  - Fung Wen Yee, Joanna
- **2005-2006 FYP students**
  - Wong Wai Yin (O'neal)
  - Ho Chiu Pun (Peter)
  - Mo Kim Tao (Laurance)
  - Wu Wan Fung (Raymond)
  - Hor Kar Chu (Laurence)
  - Ng Kwok Yuen (Jackey)

Best Paper Award in GPC2006

Terry Law (Left)   Nadia Zhang

Short Summary on Sparkle III

- Extrapolate IM usage for Pervasive Communication
  - Buddy-like interaction & awareness
- Introduce context-aware behaviors into daily application
- Separate context provision from context consumption
- Design for extensibility
- Prototype for real life usage

Conclusion

- "*Technology that disappears*" is hard to achieve, but
  - A short step could make a great impact
- ***Sentient software*** is hard to develop, but techniques are all there:
  - Aspect-oriented programming (AOP), reflection, runtime weaving, and various other adaptation techniques
  - Context Models : Call for a dynamic approach to context modeling: ***activity theory, situation theory, mental models*** could be useful
  - How to fit them in ?

Sparkle references

Project URL: http://www.cs.hku.hk/~clwang/projects/sparkle.html

- C.L. Wang , X.L. Zhang, N. Belaramani, P.L. Siu, Y. Chow, and F.C.M. Lau, **Software Infrastructure for Context-aware Mobile Computing**, to appear in Enabling Technologies for Wireless e-Business Applications, Springer.
- Francis C.M. Lau , Nalini Belaramani, Vivien W.M. Kwan, Pauline P.L. Siu, W.K. Wing, and C.L. Wang, ``**Code-on-demand and code adaptation for mobile computing**,'' to appear in Mobile Middleware, CRC Press, 2005.
- Nalini Moti Belaramani, Yuk Chow, Vivien Wai-Man Kwan, Cho-Li Wang, and Francis C.M. Lau, ``**A Component-based Software Architecture for Pervasive Computing**,'' Intelligent Virtual World: Technologies and Applications in Distributed Virtual Environments, chapter 10, pp. 191-212, World Scientific Publishing Co., Release: 07/31/2004.
- Wai-Kwong Wing, Francis Chi-Moon Lau, and Cho-Li Wang, "**Smart Retrieval and Sharing of Information Resources based on Contexts of User-Information Relationships**", *The First International Workshop on Ubiquitous Smart Worlds*, 2005
- Pauline P. L. Siu, C. L. Wang, and F. C. M. Lau, ``**Context-aware State Management for Ubiquitous Applications**,'' EUC2004.
- Laurel C. Y. Kong, C. L. Wang, and F. C. M. Lau, ``**Ontology Mapping in Pervasive Computing Environment**,'' EUC 2004.
- Yuk Chow, Wenzhang Zhu, Cho-Li Wang, Francis Chi-Moon Lau, ``**The State-On-Demand Execution for Adaptive Component-based Mobile Agent Systems**,'' ICPADS 2004.
- Vivien Wai-Man Kwan, Francis C.M. Lau, and Cho-Li Wang, "**Functionality Adaptation: A Context-Aware Service Code Adaptation for Pervasive Computing Environments**", Web Intelligence 2003.

Thanks !!

Acknowledge efforts from the Systems Research Group