Users’ Practices and Software Qualities: a Dialectical Stance

Alessandro Pollini, PhD
Interaction Design Area, Communication Science Dpt., University of Siena
pollini@media.unisi.it
Software Architecture & Usability – Bass, Kazman, John, Golden, Bosch

Empirical Investigation of Service Oriented Architecture

Architectural decisions / Use opportunities

Users’ Practices and Software Qualities: a Dialectical Stance – A. Pollini
Active Surfaces

Users’ Practices and Software Qualities: a Dialectical Stance – A. Pollini
Users' Practices and Software Qualities: a Dialectical Stance – A. Pollini
Active Surfaces

SCALABILITY

resource constrained embedded devices

number of modular networked devices
Achievements

- Future Scenarios about UbiComp and Users with Special Needs Explored and Evaluated

- PalCom Software Architecture Evaluated in Resource Constrained Devices

- Architectural Qualities Explored and Revised from both Conceptual and Operational perspectives
ASSEMBLABILITY
Manual or automatic assembly and disassembly of resources into composite constructs. Heterogeneous as well as homogeneous devices are assembled in different formations, the Assemblies that arise from the complex interconnection of devices and services in UbiComp. Assemblability provides the construction and deconstruction of services, components and devices that fits into a common model.

ADAPTABILITY
Computational resources are brought into a functional aggregate, the Assembly, the composition of which can vary dynamically. Dynamic resource reconfiguration and system behaviour modification can be effected by either programmatic autonomous means or through human interaction.

RESOURCE AWARENESS
Finding and discovering the available resources. Resources are aware of one another’s presence, availability and behaviour. Control the state of each component, capture possible faults, anticipate failures and prevent system breakdown. 1st Order (Hardware) and 2nd Order (Software) resource awareness.

EXPERIMENTABILITY
Encouraging experimentation by the users. Coherent collection of resources to be used, customised and altered within established degrees of freedom and constraints, such as performance and security.
Water supports the body and takes the weight off the joints

Movements are easier and less painful

Dealing with dynamic situation and changing conditions

Creative invention of games, in particular Symbolic Play and Physical Training
<table>
<thead>
<tr>
<th>Field Studies</th>
<th>Activity Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity Modelling</td>
<td>Definition of Requirements</td>
</tr>
<tr>
<td><strong>Activity Scenarios</strong></td>
<td></td>
</tr>
</tbody>
</table>

Water supports the body and takes the weight off the joints.

Movements are easier and less painful.

Dealing with dynamic situation and changing conditions.

Creative invention of games, in particular Symbolic Play and Physical Training.
### Field Studies
- Activity Analysis
- Activity Modelling
- Definition of Requirements
- Activity Scenarios

Water supports the body and takes the weight off the joints.

** Movements are easier and less painful.  
** Dealing with dynamic situation and changing conditions.

** Creative invention of games, in particular Symbolic Play and Physical Training.

### KEY ISSUES RELATED TO THE ACTIVITY
- Pace, Arousal and the need for Rest
- Task Duration
- Adaptation
Pace, Arousal, Activation and the need for Rest

Adaptation to different patients

General model
Pace, Activation and Rest

Pace, Arousal, Activation and the need for Rest

Adaptation to different patients

Severe physical impairments
Pace, Arousal, Activation and the need for Rest

Adaptation to different patients

![Graph showing arousal and time for hyperactive children]

Users' Practices and Software Qualities: a Dialectical Stance – A. Pollini
Tuning and Adaptation

Adaptation
Reiteration
Growing Complexity

Fine Tuning (within 1–2 min)
Contingent feedback (within 2 sec)
Overall performance (up to 30 min)
Users’ Practices and Software Qualities: a Dialectical Stance – A. Pollini

Prototyping
Prototyping

Users' Practices and Software Qualities: a Dialectical Stance – A. Pollini

<table>
<thead>
<tr>
<th>Wizard of Oz</th>
<th>Concurrent Architectural and Application Prototyping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Examples &amp; UML modeling</td>
<td>Prototype and Qualities Scenarios</td>
</tr>
<tr>
<td>Experiments</td>
<td></td>
</tr>
</tbody>
</table>
## Prototyping

<table>
<thead>
<tr>
<th>Wizard of Oz</th>
<th>Concurrent Architectural and Application Prototyping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Examples &amp; UML modeling</td>
<td>Prototype and Qualities Scenarios</td>
</tr>
<tr>
<td>Experiments</td>
<td></td>
</tr>
</tbody>
</table>

### Concurrent Prototypes Development

**Embedded Lightweight Prototype**

- **Run Time Environment**
  - **Core**
    - Introspection
    - Communication
    - Process & Thread

**Simulated Architectural Prototype**

**Embedded Architectural Prototype**

- **Execution Platform**
  - Operating system
  - Hardware

- **Runtime Engine**
  - **Process & Thread**
  - **JRE**
  - **JavaVM**

- **Middleware Management and Frameworks**
  - **Assembly**
  - **Device**

- **Utilities**
  - **Display**
  - **Storage**

---

*Users’ Practices and Software Qualities: a Dialectical Stance – A. Pollini*
Prototyping

Users’ Practices and Software Qualities: a Dialectical Stance – A. Pollini
Non-Architectural Prototype that expresses some of the core objectives of the infrastructure. Proof-of-concept application that succeeds in Demonstrating Compelling New User Experiences.

The Empirical Observation of Real Use of the Lightweight Prototype proved the Application’s Significance, still Not Probing early the Realistic Use of the Final System.

This Prototype has been Introduced in a Structured Practice and has been Explored toward Therapeutic Objectives.
Swimming Pool User Testing

Non-Architectural Prototype that expresses some of the core objectives of the infrastructure. Proof-of-concept application that succeeds in Demonstrating Compelling New User Experiences.

The Empirical Observation of Real Use of the Lightweight Prototype proved the Application’s Significance, still Not Probing early the Realistic Use of the Final System.

This Prototype has been Introduced in a Structured Practice and has been Explored toward Therapeutic Objectives.
Swimming Pool User Testing

Non-Architectural Prototype that expresses some of the core objectives of the infrastructure. Proof-of-concept application that succeeds in Demonstrating Compelling New User Experiences.

The Empirical Observation of Real Use of the Lightweight Prototype proved the Application’s Significance, still Not Probing early the Realistic Use of the Final System.

This Prototype has been Introduced in a Structured Practice and has been Explored toward Therapeutic Objectives.
Test the Potentials of the Architecture within the Simulated Active Surfaces running on a Desktop Machine.

Protected and Simplified Environment wherein aspects of the Use of Assemblies are evaluated.

How Assemblies might be rendered Easy-to-use and Understandable for the End-Users.
User Testing with the Simulator

Test the Potentials of the Architecture within the Simulated Active Surfaces running on a Desktop Machine.

Protected and Simplified Environment wherein aspects of the Use of Assemblies are evaluated.

How Assemblies might be rendered Easy-to-use and Understandable for the End-Users.
User Testing with the Simulator

Test the Potentials of the Architecture within the Simulated Active Surfaces running on a Desktop Machine.

Protected and Simplified Environment wherein aspects of the Use of Assemblies are evaluated.

How Assemblies might be rendered Easy-to-use and Understandable for the End-Users.

ASSEMBLABILITY
ADAPTABILITY
RESOURCE AWARENESS
EXPERIMENTABILITY

→ Playing with Assemblies
→ Creation and dynamic changes of assemblies
→ Game management
→ Exploration
Users' Practices and Software Qualities: a Dialectical Stance - A. Pollini

Lab Performance Testing
Users' Practices and Software Qualities: a Dialectical Stance - A. Pollini

Lab Performance Testing
Prototype based Architectural Experimentation
Performance testing based on a user-oriented perspective

Evaluation of the Architecture
- Simulation-and-prototype based approach (Bosch 2000)
- Scenario-based approach (Kazman et al. 1994)

Architectural Qualities are measured at run-time

**Observable and Quantifiable Arguments for Experimenting with Qualities**
Prototype based Architectural Experimentation
Performance testing based on a user-orientated perspective

Evaluation of the Architecture
- Simulation-and-prototype based approach (Bosch 2000)
- Scenario-based approach (Kazman et al. 1994)

Architectural Qualities are measured at run-time

Observable and Quantifiable Arguments for Experimenting with Qualities

ASSEMBLABILITY
RESOURCE AWARENESS

ADAPTABILITY

EXPERIMENTABILITY

Communication and discovery
Re-configuration
Performance
USER AND ACTIVITY REQUIREMENTS AS BASELINE FOR THE EXPERIMENTS
Time responses, delays or frequency of errors have been observed with respect to the requirements coming from the activity analysis:
  the duration of the whole session (45 minutes),
  the pace of the interaction (cycles of 3 to 5 minutes games to the utmost)
  the rests (2–3 minutes).

TASKS
12 tasks related to the Qualities and to horizontal Scalability
Comparison between communication tasks (involving IR modules) and program based tasks
Comparison between Over Time Series and Re-Boot Series
Comparison among different releases (debug and final versions) of the PalVM

OBJECTIVES
Restrictions in the PalCom open architecture or constraints due to the current hardware implementation (e.g IR communication implemented over serial port).

Overall performance of the dedicated PalComVM (PalVM)
**COMMUNICATION AND DISCOVERY**

Two series of gathered data (Re-boot and Over Time series)  
Two main actions (Put Together and Put Apart)  
Scalability factor represented by the number of tiles utilized (2, 3 or 4 tiles)

### Results – Communication and Discovery

<table>
<thead>
<tr>
<th></th>
<th>2 Tiles</th>
<th>3 Tiles</th>
<th>4 Tiles</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Intermediate VM-debug</strong></td>
<td>9.7</td>
<td>19.2</td>
<td>12.6</td>
</tr>
<tr>
<td><strong>Final VM-debug</strong></td>
<td>5.25</td>
<td>6.5</td>
<td>6.8</td>
</tr>
<tr>
<td><strong>VM release</strong></td>
<td>3.2</td>
<td>7</td>
<td>6.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>2 Tiles</th>
<th>3 Tiles</th>
<th>4 Tiles</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Intermediate VM-debug</strong></td>
<td>11.5</td>
<td>13.44</td>
<td>19.3</td>
</tr>
<tr>
<td><strong>Final VM-debug</strong></td>
<td>11</td>
<td>13</td>
<td>14.55</td>
</tr>
<tr>
<td><strong>VM release</strong></td>
<td>7.6</td>
<td>9.8</td>
<td>12.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>2 Tiles</th>
<th>3 Tiles</th>
<th>4 Tiles</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Intermediate VM-debug</strong></td>
<td>13.5</td>
<td>15.8</td>
<td>16.2</td>
</tr>
<tr>
<td><strong>Final VM-debug</strong></td>
<td>5.05</td>
<td>8.7</td>
<td>6.95</td>
</tr>
<tr>
<td><strong>VM release</strong></td>
<td>3.5</td>
<td>7.6</td>
<td>7.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>2 Tiles</th>
<th>3 Tiles</th>
<th>4 Tiles</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Intermediate VM-debug</strong></td>
<td>14</td>
<td>17.2</td>
<td>20</td>
</tr>
<tr>
<td><strong>Final VM-debug</strong></td>
<td>11.4</td>
<td>11.46</td>
<td>13.7</td>
</tr>
<tr>
<td><strong>VM release</strong></td>
<td>8.1</td>
<td>10.6</td>
<td>13.3</td>
</tr>
</tbody>
</table>
ASSEMBLABILITY
Manual or automatic assembly and disassembly of resources into composite constructs. Heterogeneous as well as homogeneous devices are assembled in different formations, the Assemblies that arise from the complex interconnection of devices and services in UbiComp. Assemblability provides the construction and deconstruction of services, components and devices that fits into a common model.
Assembly of Assemblies, on the lower level each device constitutes an assembly, which is part of another assembly that is made of assembled constructs at the higher level Scalability factor.
Different levels specified by hierarchical user access and ownership.

ADAPTABILITY
Computational resources are brought into a functional aggregate, the Assembly, the composition of which can vary dynamically. Dynamic resource reconfiguration and system behaviour modification can be effected by either programmatic autonomous means or through human interaction.
User adaptation as intuitive and perceivable manners of changing assemblies, as experimented with physical programming–by–example.
Gradual adaptation and pace of the adaptation.
System proactive initiative and user acceptance.
RESOURCES AWARENESS
Finding and discovering the available resources. Resources are aware of one another’s presence, availability and behaviour. Control the state of each component, capture possible faults, anticipate failures and prevent system breakdown. 1st Order (Hardware) and 2nd Order (Software) resource awareness.
Perceive and discriminate among the available and functioning resources.
Reading the behaviour of the system in a manner that is congenial for the users.

EXPERIMENTABILITY
Encouraging experimentation by the users. Coherent collection of resources to be used, customised and altered within established degrees of freedom and constraints, such as performance and security.
Experiment and trying novel configurations out, meaningful for the users, unexpected for the designers.
Field exploration, environment and physicality.
EVALUATION OF UBIQUITOUS SYSTEM
Software and hardware resources are distributed throughout the physical world and this impacts individual and social behaviours.
Different evaluation criteria user attention (focus and overhead), the adoption of the system (value and availability) and the qualities of the interaction (physically embeddedness, dynamic input/output, multiple devices, multiple users). Others are Understanding, control, accuracy, appropriateness, and customization (Scholtz, Consolvo 2004)

PERFORMING AND EVALUATING SERVICE ORIENTED ARCHITECTURES
Exemplar usability features are Using Applications Concurrently, Recovering from Failure, Reusing Information, Working at the User’s Pace and Predicting Task Duration (Bass et al. 2001)

UBICOMP FOR PEOPLE WITH SPECIAL NEEDS
Requirements from universal access and user diversity: personalization and tailoring to user needs: i.e., it can recognize the user, evolve throughout time and keep track of the history of the interactions; adaptivity: i.e., its behaviour can change in response to a person’s actions and environment; anticipatory system: i.e., it anticipates a person’s desires and environment as much as possible without mediation.
Integration among the traditional ethnographic studies, participatory design methods and naturalistic experiments to inspire, inform and evaluate the design of the software architecture.

Users’ Practices and Software Qualities: a Dialectical Stance – A. Pollini
Empirical investigation on the dialogue between user research and software development

As designers we play the mediator role between end users and software architects
Participatory design and multidisciplinary team (software engineers, interaction designers and psychologists)

Data gathered during the field exploration and the activity analysis provided the scaffold to build the experimental plan and the baseline for the evaluation of the outcomes

Activity, Envisioning, Prototype and Qualities Scenarios tested and evaluated with the users

Architectural prototypes and application prototypes to discover the gradually emergent requirements and to gather user feedback at critical time.

The introduction of prototypes enhance users’ practices by enabling novel use opportunities meanwhile users’ actions provoke, inspire and inform the emergence of unpredicted needs that require new architectural solutions.
This research has been supported by PalCom, Palpable Computing, an integrated project in FP6 ‘The Disappearing Computer in Future and Emerging Technologies’ (FET)

For further reference see: http://www.ist-palcom.org/