Extensibility as a Collaboration Enabler: A Case Study for Group-Context-Aware Mobile Applications

Marc Novakouski
Grace A. Lewis
Enrique Sánchez

Software Engineering Institute
Research, Technology and Systems Solutions (RTSS) Program
Advanced Mobile Systems (AMS) Initiative
May 9, 2012
Copyright 2012 Carnegie Mellon University.

This material is based upon work supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This material has been approved for public release and unlimited distribution except as restricted below.

Internal use:* Permission to reproduce this material and to prepare derivative works from this material for internal use is granted, provided the copyright and “No Warranty” statements are included with all reproductions and derivative works.

External use:* This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other external and/or commercial use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

*These restrictions do not apply to U.S. government entities.
Agenda

Background
Architecture Drivers and Scenarios
Architecture Decisions
Extensibility as a Collaboration Enabler — Results
Conclusions
Agenda

Background
Architecture Drivers and Scenarios
Architecture Decisions
Extensibility as a Collaboration Enabler — Results
Conclusions
Group-Context-Aware Mobile Applications

Context-aware mobile applications are capable of sensing and responding to changes in their environment or context.

Group-context-aware mobile applications integrate the individual’s context with that of nearby individuals operating as part of a group or unit, such as in the military or first responder situations.

Integrated context is used to enhance the precision of information provided to users as well as a more complete picture of the status of a mission.

- Goal is to produce a capability that can sense as much of the emerging context as possible and apply that context to filter data such that only the most relevant information is displayed.
Desired characteristics of the solution include:

- Capturing context information on a handheld device in a non-intrusive manner
- Extending the sources of contextual information beyond location and time
- Storing context information and disseminating this information to peers
- Capturing and using context information efficiently without imposing an unreasonable burden on handheld device resources
- Integrating local and group context information and only displaying information that is of relevance to the individual and mission according to pre-defined rules
Motivation

One of the more interesting results of this work has been the ability to leverage the architecture to support collaboration. By identifying extensibility scenarios early on in the design process, we were able to construct an architecture that supports multi-organizational collaboration to construct and evaluate different pieces of the architecture:

- context data models
- context data storage
- context sensors
- context reasoning engines and rules
- context views

This has allowed us to reach out to researchers from multiple universities and industry, resulting in synergistic research and development, furthering the goals of all participants.
Agenda

Background

**Architecture Drivers and Scenarios**

Architecture Decisions

Extensibility as a Collaboration Enabler — Results

Conclusions
Business and Architectural Drivers

Business Drivers

- Opportunistic integration of new technology
- Ease of integration with components produced by collaborators
- Applicability of architecture to different edge-enabled applications

To meet business drivers we defined **extensibility** as the main architectural driver.
## Extensibility Scenarios

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Attribute Concern</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Add a New Sensor</td>
<td>Separation of Concerns</td>
</tr>
<tr>
<td>2</td>
<td>Add a New Sensor</td>
<td>Modifiability</td>
</tr>
<tr>
<td>3</td>
<td>Add a New Communication Mechanism</td>
<td>Separation of Concerns</td>
</tr>
<tr>
<td>4</td>
<td>Add a New Communication Mechanism</td>
<td>Modifiability</td>
</tr>
<tr>
<td>5</td>
<td>Add a New Context Event / Action</td>
<td>Separation of Concerns</td>
</tr>
<tr>
<td>6</td>
<td>Add a New Context Event / Action</td>
<td>Modifiability</td>
</tr>
<tr>
<td>7</td>
<td>Add a New Context View</td>
<td>Separation of Concerns</td>
</tr>
<tr>
<td>8</td>
<td>Add a New Context View</td>
<td>Modifiability</td>
</tr>
</tbody>
</table>
## Scenario 3: Add a New Communication Mechanism

<table>
<thead>
<tr>
<th>Scenario Refinement</th>
<th>Attribute</th>
<th>Concern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add a New Communication Mechanism</td>
<td>Extensibility</td>
<td>Separation of Concerns</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stimulus</th>
<th>Source</th>
<th>Environment</th>
<th>Artifact</th>
<th>Response</th>
<th>Response Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td></td>
<td>Developer is sufficiently comfortable with application to make changes in a reasonable amount of time</td>
<td>Communications Manager of the context-aware system</td>
<td>Communications Manager is changed to implement message passing using the new communication mechanism</td>
<td>Aside from communication-mechanism-specific code, only the Communications Manager is changed to accommodate the new communications mechanism.</td>
</tr>
</tbody>
</table>
### Scenario 5: Add a New Context Event / Action

<table>
<thead>
<tr>
<th>Scenario Refinement</th>
<th>Attribute</th>
<th>Concern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add a New Context Event/Action</td>
<td>Extensibility</td>
<td>Separation of Concerns</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scenario Refinement</th>
<th>Stimulus</th>
<th>Source</th>
<th>Environment</th>
<th>Artifact</th>
<th>Response</th>
<th>Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stimulus</td>
<td>Developer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source</td>
<td>Developer identifies a new event that can be detected by examination of context data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Environment</td>
<td>Developer is sufficiently comfortable with application to make changes in a reasonable amount of time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Artifact</td>
<td>Context Engine of the context-aware system</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Response</td>
<td>Context Engine is changed to detect the conditions for the event and generate a new action when it is detected</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Response Measure</td>
<td>Only the Context Engine is changed to allow for detection of events and generation of actions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Agenda

Background
Architecture Drivers and Scenarios
Architecture Decisions
Extensibility as a Collaboration Enabler — Results
Conclusions
Architectural Decision 1: Communications Interface

Challenge: Integration of very different communication mechanisms

• Different protocols support different use cases
• Target hardware is unknown
• Need to adapt to target network capability

Solution

• Common service interface provides generic communication methods and callbacks that individual protocols can adapt as necessary
• Allows any sequence of communication events to account for differences in protocols
Architectural Decision 2: Sensor Interface

Challenges

• Integration of any current or future available sensor
• Control of sample rate and change threshold

Solution

• Common sensor interface provides generic communication methods that individual sensor implementations can adapt to as appropriate
Problem

Peer review of the architecture raised the issue that a single thread would cause high-frequency sensors to overwhelm the application.

Simple experimentation demonstrated that this was indeed a problem.

Solution

- Sensors implemented as Android Services (processes separate from the application)
- Communication via IPC to insulate application from high poll rate impact

Tradeoff

- Higher complexity in sensor implementation although interface hides as much as possible.
Architectural Decision 3: Context Model “At the Center”

Challenges

• Easy creation of rules based on contextual data captured via sensors or user input
• Standardized rule processing

Solutions

• Generic and extensible context model that can handle a wide range of situations, environments, data
• Standardized rule set read by application from XML file

Tradeoff

• Both sensors and views have to know the context model element that they are affecting — strong coupling

Simple example of rules that would codified in XML

If (BATTERY_LEVEL=85) then createAlert("BATTERY LOW")

If (WATER_SUPPLY=LOW) then
peer = findPeer(MAX(WATER_SUPPLY) and (MIN(distance)))
createMessage(peer + " has water. Location = " + peer.LOCATION)
Architectural Decision 4: Standardized Messaging

Challenge

• Easy creation of views that can capture and/or display context data

Solution

• Publish/subscribe interface
  – Standardized set of actions that can be created by the context engine as the result of fired rules
  – Application manager publishes actions created by context engine as standardized events
  – Views subscribe to events
First Responder Application Architecture
Agenda

Background
Architecture Drivers and Scenarios
Architecture Decisions
Extensibility as a Collaboration Enabler — Results
Conclusions
Results

The extensible architecture enables productive collaboration

• Sensor and communication service interface enable 3rd parties to contribute new/novel sensor and protocol implementations
• Standardized rule set approach allows adaptation to different context data models
• Standardized messaging enables easy integration of new context data views
Results

Collaborators at GMU were able to modify their unique communication protocol to interface with application architecture in just a few weeks.

Collaborators are working on developing a group context data model, unconstrained by implementation details and without affecting our progress in the meantime.

Collaborators within SEI planning to integrate related projects for QoS management, code offloading, and end-user programming with no foreseen complications.
Agenda

Background
Architecture Drivers and Scenarios
Architecture Decisions
Extensibility as a Collaboration Enabler — Results

Conclusions
Conclusions

Extensibility as an architecture driver enables productive collaborative research and development

Scenario-driven architecture design along with peer architecture evaluation is useful even for small projects
  • Concrete definition of quality attribute requirements
  • Early identification of risks and tradeoffs
Contact Information

Marc Novakouski
Research, Technology and Systems Solutions (RTSS) Program
Advanced Mobile Systems (AMS) Initiative

Software Engineering Institute
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

Phone: +1 412-268-4274
Email: novakom@sei.cmu.edu
WWW: http://www.sei.cmu.edu/staff/novakom