Nokia Acquire Engine

Technical Report

Prepared By,
Chris Brodhecker
Josh Gruenberg
Mehlam Kalverts
Matthew Mason

Faculty Coach: Prof. Raghu Reddy

Last Revised: 5/13/2009
# Table of Contents

Revision History .................................................................................................................. 3

I. Project Overview ................................................................................................................. 4

II. Basic Requirements ........................................................................................................... 5

III. Constraints ......................................................................................................................... 6

   a) Maven ............................................................................................................................... 6
   b) GWT Compiler .................................................................................................................. 6
   c) OBEX Object Push Profile ............................................................................................... 6
   d) MMS Service .................................................................................................................... 7
   e) Team Schedule .................................................................................................................. 7
   f) Testing Hardware .............................................................................................................. 7

IV. Development Process ....................................................................................................... 8

V. Project Schedule ............................................................................................................... 9

VI. System Design ................................................................................................................... 11

   a) Overall System Architecture ......................................................................................... 11
   b) Content Delivery ............................................................................................................ 13
   c) User Interface Design ..................................................................................................... 15

VII. Process and Product Metrics ........................................................................................ 17

VIII. Product State at Time of Delivery .............................................................................. 19

IX. Project Reflection ............................................................................................................. 20

X. Resources .......................................................................................................................... 21
## Revision History

<table>
<thead>
<tr>
<th>Name</th>
<th>Date</th>
<th>Reason For Changes</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Draft</td>
<td>May 11, 2009</td>
<td>Initial Compilation</td>
<td>1.0</td>
</tr>
<tr>
<td>Revision 1</td>
<td>May 12, 2009</td>
<td>Added Process and Product Metrics</td>
<td>1.1</td>
</tr>
<tr>
<td>Revision 2</td>
<td>May 13, 2009</td>
<td>Grammatical changes, formatting, diagram numbering.</td>
<td>1.2</td>
</tr>
</tbody>
</table>
I. Project Overview

The Nokia Acquire Engine is a rich web application, focusing on providing content to users through Bluetooth technology. Specifically, the application consists of a client interface, in the form of a web application, a Bluetooth server that is tasked with discovering local Bluetooth enabled devices, and a centralized database that will provide necessary information, updates, and data storage to the deployed client applications. In addition to providing a client interface, the system has an administrative interface that can remotely interact with the central database. The administrative interface provides functionality to create new "campaigns" and assign them to specific kiosks. The client interface, accessible via a kiosk or standard web browser, allows users to view available campaigns and request content. Additionally, usage statistics are kept detailing a user's interactions with the system, including kiosk interactions, page views, and content transfers initiated.
II. Basic Requirements

There are three major components to the Nokia Acquire Engine, the Bluetooth server, kiosk/web user interface, and administration interface.

The Bluetooth server searches for nearby Bluetooth enabled devices and offers to send content to each capable device. Examples of such content are free wallpapers, ringtones, or videos. The application should be easy to localize for different languages, since Nokia is an international company. Initially, this application will be for internal use within Nokia. Concerns that are necessary to consider for public deployment such as security and privacy are not an issue for the scope of this project. A reliable, high speed internet connection is also assumed.

The user interface is accessible from a kiosk or any web browser. It will list the currently available content and allow users to request the content via e-mail, MMS, or Bluetooth, if they are using a kiosk. This website has a rich, application-like feel. In addition to supporting Bluetooth sends from kiosks, the web interface has a "time out" function, in the case that a user leaves the send process when it is halfway completed. By the time another user approaches the kiosk, the kiosk should have reset to the home screen. In addition to providing content, users can also sign up to receive SMS or e-mail alerts about new content.

The administration interface allows Nokia to add content and set when it should be available, as well as view usage statistics. Statistics can include the number of times that content is sent, the number of unique devices seen per kiosk, as well as the most popular protocol by which to send files. Campaigns can also be assigned to specific kiosks to be anonymously bluecasted, but a kiosk may only have one bluecast assigned to it at a time.
III. Constraints

a) Maven
Apache's Maven is a dependency management and project build tool similar to Ant. However, Maven introduces the idea of structure over specification. Instead of explicitly specifying where the source code, configuration resources, and dependencies are located, Maven expects them to exist within a certain structure. For example, all source code will exist in `/src/main/java` while all resources exist in `/src/main/resources`. This is useful in that it greatly simplifies the Maven configuration files. For more information on Maven and its structure, see the relevant link in the “Resources” section.

b) GWT Compiler
The only significant constraint that had influence on the project's architecture was that of the Google Web Toolkit (GWT) Java to JavaScript compiler. GWT works by taking Java classes with approved UI elements and compiling them into JavaScript. All classes to be compiled by the GWT compiler needed to be placed within the com.nokia.gwt.client package. Also any external classes used by the UI, such as that in the common project or third party libraries, needed to have their source code available to the compiler.

In addition to the project structure, the GWT compiler limits the amount of functionality on the client side. The GWT compiler cannot serialize all classes. Because of this, the only custom classes communicated between the server and clients were simple data classes. These data classes held little more than primitives and Java collections, along with associated getters and setters.

c) OBEX Object Push Profile
The OBEX object push profile is a Bluetooth protocol that allows devices to receive content anonymously without the need to pair devices that need to communicate. The requirement from our sponsors was to be able to broadcast content without the need to explicitly pair devices. This is why the system uses this protocol to send out content to phones. Using this profile, however, restricted the variety of devices that our system could work with. For the system to be able to successfully push out content, the receiving device would not only need to have Bluetooth enabled, but would also need to implement the OBEX object push profile. We found that several Bluetooth enabled devices detected by our system did not have this profile implemented and would not be able to receive our content.
d) MMS Service
A project requirement was to be able to deliver content via MMS. Initially, the requirement involved users being able to request specific content to be sent to their phones. The user would send a specific message to a service number which would then respond via MMS with the requested content. However, Nokia did not have an existing MMS server that could handle incoming messages and respond to them. The team believed that writing such a service from the ground up was not within the scope of the project and could not be delivered in a timely manner.

An alternative solution was proposed to Nokia and settled on. In order to send content to devices via MMS the system would email the content to a vendor specific email addresses from where they would be forwarded to the intended recipient. For ex., if some content has to be sent out to a Verizon customer with the cell number 400-500-6000 then the system would generate an email with the content and send it to 4005006000@vzwpics.com from where Verizon would forward the content to the intended device via MMS. This method allowed the system to achieve some of the MMS functionality without it being written from the ground up.

e) Team Schedule
Another development constraint was the varied schedule of the developers. There was significant difficulty in scheduling meetings with all team members to collaborate on development and to problem solve. This made the turnaround on tasks and overall development effort a little slower than the team would have liked. This was because the team ran into several challenges and difficulties while developing using new technologies such as GWT, Maven, and the Spring Framework. These obstacles may have been overcome a little quicker if more frequent meetings could have been scheduled.

f) Testing Hardware
The interaction with hardware made testing during the development of this project challenging. Key features such as bluecasting, kiosk Bluetooth delivery, and MMS content delivery, could only be tested with Bluetooth enabled devices. Nokia sent over two Nokia E66 phones to test with which proved invaluable. However, testing system compatibility with other device manufacturers and various cell phone providers was sporadic.
IV. Development Process

Due to the use of unfamiliar technologies and the associated risk, the spiral methodology was chosen to develop the Acquire Engine. Nokia had no specific preference regarding methodology, but wished to determine the user interface as a top priority. The spiral allowed us to deliver prototypes of the system every four weeks. The incremental prototype system also took into account the communication difficulties that come with remote sponsors by serving as visible signs of progress for both Nokia and our team. The spiral also made it easy to determine how closely the project was adhering to the planned schedule.

The project was broken up into five distinct spirals. The delivery of content through Bluetooth was deemed the riskiest and most unfamiliar part of the project and was placed in the first spiral. The user interface was also placed in the first spiral in order to satisfy Nokia's wish to quickly determine the look and feel of the product.

The second spiral consisted of determining the layout and design of the administrative side of the Acquire Engine and implementing the aspect oriented statistics collection framework.

In spiral three the layouts of both the administrative and user interfaces were aesthetically styled. The styling was put off until the third spiral in order to give time to Nokia to evaluate the layouts created in the previous spirals.

Spiral four saw the integration of Bluetooth, MMS, and email content delivery with the user interface. The styled interfaces were connected to the databases as well. The prototype created from this spiral contained all the general functionality requested.

The mobile user interface was a low priority relative to the rest of the system and thus placed in the final spiral in case of project delays. This feature has been scoped out due to time constraints and has a low impact on the overall project. Usability testing was planned for this spiral as well.
V. Project Schedule

The schedule for this project was put together based on the methodology chosen (spiral) and by looking at the most important deliverables.

Initially, four month long spirals running from January – May were identified each with individual components of the system being worked on. As tasks were more clearly identified and greater prioritization was achieved, development was broken down into five spirals of 2-4 weeks.

The first and second spirals were used to deliver user interface prototypes for the client and administration user interfaces respectively. These spirals had a small amount of slippage due to the fact that we were learning how to design user interfaces in GWT. Additionally, during the first spiral, the team worked on prototyping a Bluetooth server that would have the ability to push content out to OBEX enabled devices.

The third spiral evolved into a phase devoted solely to styling the user interfaces. We had noticeable slippage here as the team as a whole had very little design experience. Even though we had user interfaces ready from a functional standpoint, they were not visually appealing. Applying CSS styling while working with GWT proved very tricky and consumed much more time than estimated.

The fourth spiral was dedicated to hooking up the user interface to the database and populating it real data. Additionally, during this spiral, the team worked on implementing the MMS content delivery system.

The fifth spiral was intended to be where a light-weight mobile user interface was created. This would be the interface that devices would access while on the go where bandwidth and fully functional browsers may not be available. This was scoped out because of overrun from the third spiral and due to the fact that two user interfaces had already been designed. This interface had been a low priority from the beginning and therefore had minimal impact to the rest of the project. Some time allocated for the fifth spiral was intended for testing which was carried on as planned.

The team was flexible with the schedule since there were known risks upfront. The two interfaces which were of high priority and demanded working with new technologies were scheduled upfront so as to allow time for feedback from the sponsor and to minimize the risk as a result. Additionally, when it became apparent through discussions with the sponsor that an
MMS server would not be provided, the team mitigated the risk by pushing development of that subsystem into a future spiral when clearer requirements would be available. This flexibility allowed us to work on components of the system that were fully defined first while other requirements were being clarified.
VI. System Design

a) Overall System Architecture

As shown in the above diagram, there are two main aspects to the Bluecast system: a centralized server and remote kiosks. The centralized server is where the two user interfaces, the client and the administrative, are deployed. The various remote kiosks each have their own instance of the Bluetooth server. By deploying the user interfaces as web applications on a centralized server, users and administrators can access them anywhere that has an internet connection, including the kiosks. Also, by concentrating them onto one server, deploys and upgrades are greatly simplified.

The kiosks interact with the centralized server in two ways. First, the Bluetooth server is started
on the kiosk. The Bluetooth server contains the address of the centralized server in a configuration file. It will attempt to establish a socket connection with the centralized server. When this connection is successfully established, the kiosk will receive a kiosk ID from the centralized server. It will then write out a file containing the URL of the client application with the kiosk ID added as an argument. This begins the second interaction with the centralized server. Whoever is deploying the Bluetooth server will then open an internet browser and paste the URL in the address bar. Since the URL contains the registered kiosk’s ID, the centralized server will know where the content requests are coming from. When it receives a request to push content over Bluetooth from a kiosk, it will look up the kiosk’s ID in its mapping of socket connections and push the content to the Bluetooth server. This, in turn, will then push the content to the mobile device.

It was discussed whether the project should use HTTP or sockets to transfer data requests from the centralized server to the Bluetooth server. It was decided that sockets would be simpler for the scope of the project, as it would avoid the need to implement HTTP security. Although security was not a high priority for the project, it was important here. The client (Nokia) did not want all their content exposed via HTTP, allowing an end user to circumvent the Bluecast system and just pull data through HTTP request. Sockets provided a simple way to encapsulate data transfer and enforce use of the system.
b) Content Delivery

At the heart of the Bluecasting system is its content delivery system. The above diagram demonstrates how this is handled within the system. Basically, protocol responsible for transporting content to the client will implement the TransferService interface. This has one method, `send(String destination, Campaign campaign)`, that specifies a destination and the campaign to send to it. A destination can be an email address or a Bluetooth OBEX URL. If the scope of the project is increased in the future and a new protocol needs to be supported, it need only have a TransferService implementation.

Bluetooth transfers use the OBEX Object Push Profile (OPP) to send to discovered devices anonymously, without needing to pair with the phone initially. However, for security reasons, many phones do not allow users to receive content without first pairing with the sending device using a secure pass code. Our service queries the device to see if the profile is supported, and if so, initiates a bluecast. It has been found from our testing that the profile is implemented
correctly on Nokia and Sony Ericsson devices. It does not work with the iPhone, but does appear to work with some MacBook laptops. It has not been found to work on any phones on Verizon. An LG phone on AT&T was tested, but despite the phone reporting that it supported the correct profile, the phone insisted on pairing with the sending device first, making it a false positive.

A requirement of the system was support sending content to devices through the Multimedia Message Service (MMS), or picture text messages. Originally it was discussed that a messaging framework be used to communicate with devices. This, however, proved to be not feasible. Instead, it was determined that the project would exploit the emailing of content to phones. This is possible as every carrier defines an email address for a phone. This is typically in the form of phone_number@carrier.com. When requesting content through MMS, the user is prompted for their phone number and carrier. The service then computes the correct email address, creates the message, and then adds the content in the form of attachments. This allowed for a lot of code reuse between the EmailTransferService and the MmsTransferService.
c) User Interface Design

Asynchronous calls to the server from the client force a slightly non-traditional user interface system design. If a user clicks on a campaign’s page, the system needs to make a call to the server to get the campaign’s details. Since this call is asynchronous, in order to ensure that the page changes only when the data is available, the response to the client must make a call to the central class to change the page. This is represented by the `setContent(DisplayablePage displayablePage)` method in `Client_UI`. When a call successfully comes back from the server, the content panel is set to a `DisplayablePage` implementation. The method takes an instance of the page and gets its content by invoking its `getPanel()` method. This model is what drives both the client and administrative user interfaces.
Additionally, due to GWT’s lack of history management, a class had to be created to handle the back and forward buttons on browsers. Otherwise, the browser would treat the whole application as one page, thus removing a user’s ability to navigate between pages with the back and forward buttons. This functionality is located in the HistorySupportedPage.
VII. Process and Product Metrics

The Acquire Engine's goal is to supply a fun and rewarding experience to the user. In order to achieve this, the application must be reasonably responsive in order to keep the user interested. This is especially important in a kiosk setting where the user can step away from the experience at the first sign of frustration. To this end we decided to evaluate application responsiveness by measuring the time it takes for the database to serve a request. The measurements were taken from the user's perspective: the time from when the mouse click is made to the desired outcome.

Our measurements showed the average response time to be 0.71 seconds. This measurement met our goal of interactions taking less than one second. A closer look at the data revealed that, although acceptable, this average was being skewed by a number of outliers. Removing the outliers resulted in an even more desirable average of 0.452 seconds. The standard deviation of the measurements was 0.129 seconds. The outliers were all over three standard deviations away. While these outliers are a concern their numbers were not large enough to justify a closer inspection of their cause.

It is recommended that the measurements be taken again when the system is transferred to Nokia's control on their databases and on a regular basis as the amount of data in the database increases to ensure that the system continues to be adequately responsive.

We tracked our progress, or lack thereof, by employing a slippage chart. This allowed Nokia to easily evaluate our current status, overall progress, and momentum. The slippage chart was also used internally to determine when scheduling changes or scope reductions needed to take place. Figures 4-9 show the project slippage. The figures are drawn from the master slippage chart on the project website. They have been split up here to increase readability.

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Start</th>
<th>Finish</th>
<th>Sun Nov 16</th>
<th>Thu Nov 27</th>
<th>Mon Dec 8</th>
<th>Fri Dec 19</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - Setup Phase</td>
<td>Mon 12/1/08</td>
<td>Fri 12/10/08</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 - Initial virtual machine setup</td>
<td>Mon 12/1/08</td>
<td>Wed 12/17/08</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 - Setup project website</td>
<td>Mon 12/3/08</td>
<td>Fri 12/19/08</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Figure 4 - Setup Phase]
### Spiral 1
- **Mon 1/5/09**
- **Mon 1/26/09**
- **Tue Dec 30**
- **Sat Jan 10**
- **Wed Jan 21**
- **Sun Feb 1**
- Research bluetooth server technologies
- Mon 1/5/09
- Mon 1/12/09
- Implement bluetooth server
- Mon 1/12/09
- Tue 1/20/09
- Complete page layout for client user interface
- Mon 1/5/09
- Mon 1/26/09

![Figure 5 - Spiral 1](image1.png)

### Spiral 2
- **Tue 1/27/09**
- **Fri 3/6/09**
- **Wed Jan 21**
- **Sun Feb 1**
- **Thu Feb 12**
- **Mon Feb 23**
- **Fri Mar 8**
- Complete page layout for admin user interface
- Tue 2/2/09
- Tue 2/24/09
- Styling of client user interface
- Tue 1/27/09
- Fr 3/6/09

![Figure 6 - Spiral 2](image2.png)

### Spiral 3
- **Tue 3/3/09**
- **Mon 4/6/09**
- **Mon Feb 23**
- **Fri Mar 6**
- **Tue Mar 17**
- **Sat Mar 28**
- Styling of admin user interface
- Tue 3/3/09
- Tue 3/24/09
- Configure individual builds for projects
- Tue 3/3/09
- Thu 3/5/09
- Setup localization framework
- Tue 3/3/09
- Wed 3/4/09
- Setup spring AOP
- Mon 3/9/09
- Fri 3/13/09
- Implement statistics collection for bluetooth server
- Thu 3/12/09
- Mon 4/6/09

![Figure 7 - Spiral 3](image3.png)

### Spiral 4
- **Tue 4/7/09**
- **Tue 5/5/09**
- **Sat Mar 28**
- **Wed Apr 8**
- **Sun Apr 19**
- **Thu Apr 30**
- Connect client user interface to database
- Tue 4/7/09
- Fri 4/24/09
- Connect admin user interface to database
- Tue 4/7/09
- Tue 5/5/09
- Implement small SMS system
- Mon 4/13/09
- Fri 4/17/09

![Figure 8 - Spiral 4](image4.png)
VIII. Product State at Time of Delivery

The Bluetooth server will run, register itself with the database, provide the unique URL to the corresponding kiosk, and begin doing its work. This includes scanning for nearby devices and updating the database, bluecasting the current campaign to any devices that have not received or declined it before, and executing any sends initiated from the client user interface. It has been tested on Microsoft Windows using Microsoft's Bluetooth stack, and should function with Broadcom and BlueSoleil's stacks as well, but has not been tested. It has also been run on Linux with the BlueZ stack. Extensive testing with multiple Bluetooth kiosks and running two Bluetooth servers on computers within range of each other has not been carried out.

The client user interface is fully styled and contains content viewing, rating, time out, and Bluetooth, MMS, and email sending features. It has been fully tested in Mozilla Firefox 3, but may require some work to tweak appearance in Internet Explorer and other browsers. The MMS feature currently uses carrier-provided e-mail addresses to send files, before moving to a production environment it may be necessary to change this to use custom functionality to ensure that the message is presented in the best possible format. Our architecture accounts for this possibility with the TransferService interface explained above.

At the time of writing, functionality of the administration user interface is still being developed. Administration pages are created and styled, and some are connected to the database. However, working with GWT's methods for handling uploaded files has proven to be very difficult. When complete, functionality will include creating new campaigns, adding files to a campaign, assigning campaigns to kiosks, and viewing kiosk statistics. In the case that work on file uploads cannot be completed, an alternate plan to insert files using open-source database management tools will be documented and delivered.
IX. Project Reflection

During the development of this system the team had to learn to work with several new technologies such as GWT, Maven, and the Spring Framework. This was both exciting and challenging. There was a steep learning curve that the team had to overcome but this was a rewarding process.

At several stages the team was faced with design problems. These ranged from developing swappable user interface pages to the content delivery mechanism for use via Bluetooth and MMS. The team feels that solid design decisions were made to allow for extensibility, modifiability, and maintainability of the system.

The team also noted a few things that could have been done differently to improve our efficiency. We met twice a week to discuss progress and assign new tasks. However, at times an additional meeting over the weekend would have proved useful in assigning additional tasks or to problem-solve. In retrospect, we would also break our tasks down into smaller chunks in order to better assign them to team members. This would also improve our estimation efforts and allow us to better plan the progress of the project. Due to the number of technologies being used, it was difficult to provide the sponsor with frequent builds of the system. In order to test a build, a certain amount of setup and hardware was necessary and this hampered us from creating frequent builds.

Finally, the project overall involved far more user interface / website design than the team was expecting. Significant amounts of time were set aside in order to develop interfaces to meet expectations. The lack of any support from the sponsor in this area made the task quite difficult. The team would have liked to spend more time addressing more technically challenging areas such as content delivery and system robustness.
X. Resources

Maven - http://maven.apache.org/