Javelin:
A Personal Communication Device Demo

Annette Wagner
Sun Microsystems, Inc.
901 San Antonio Road, CUP01-203
Palo Alto, CA 94303 USA
+1 408 343 1472
annette.wagner@sun.com

ABSTRACT
This paper describes the creation of a set of demonstration applications for a personal communication device for use at the JavaOne 99 Developers Conference. The major design issues encountered in the project are described along with the resulting impact on the project. These include defining the navigation model for the Back key, dealing with the issues that arose when the device hardware target was changed in mid-process, and managing the design process to leverage prior work when major goals were altered late in the project.

Keywords
Consumer electronics, collaboration, personal communication device, design, user interface, java technology, human interface, virtual machine, pager, navigation model, style guide, highlight traversal.

BACKGROUND
In October of 1998, Sun Microsystems Inc. and Motorola Inc. starting work on a project to create a demonstration personal communication device for the JavaOne 99 Developers Conference. The device, code named Javelin, was based on a two-way pager and included a variety of demonstration applications for use at the conference. (See Illustration 1 for a screenshot of the final Javelin device.) The software stack was based on a new Java (TM) virtual machine called K Virtual Machine (KVM).

The JavaOne conference is a yearly event which is attended primarily by software engineers working with Java technology. The conference includes tutorials, sessions, exhibits, and other events. It is held in San Francisco, California, at the Moscone Convention Center. Attendees are primarily Java software developers with a small percentage of managers and marketing personnel. Over 20,000 people attended JavaOne 99.

DIS '00, Brooklyn, New York.

The primary goal for the Javelin project was to show a compelling set of demonstration applications at JavaOne 99 using the KVM and other Java technology on one of Motorola's personal communication devices.

Illustration 1: A scan of the Javelin device in its completed form. The home screen is displayed with icons used for the final set of applications.

The Team
The Javelin project involved a collaborative team of people from Sun, Motorola, and the producers of the JavaOne conference, ZD Events. There were three aspects of the organization that impacted the development of the Javelin device. The JavaOne 99 conference was still being planned

as the team was developing software, consequently, constant communication was required to stay in sync with the daily changes. The teams were not co-located. The primary teams were in Cupertino, California and Austin, Texas. Others were dotted across the United States. Communication was accomplished largely by e-mail, phone calls, and some face to face meetings. Last, the software integration team in Austin had not had an opportunity to work closely with a human interface designer before and there was some education required to bring everyone up to speed and clarify roles.

A reference implementation of the KVM was being written by a team of Sun engineers. They in turn worked with key engineers at Motorola to get a version of the reference implementation running on the target hardware. A different set of Motorola engineers operating as an integration team focused on getting the rest of the stack including applications up and running. The author was the human interface person assigned to the project working with both the Motorola integration team and the KVM team at Sun.

There were key marketing people from both companies who reviewed the work and provided direction and feedback as the demonstration progressed. Finally, there was the team at Sun who worked with the conference producers. All our information about the conference came through these people.

MAJOR DESIGN ISSUES

Constraints

The major constraint on the project was to meet the deadline of showing the Javelin device at JavaOne 99. The conference ran from June 14th through the 18th. Software had to be completed early enough to allow time for downloading the software onto the devices going to the conference.

The choice of device also introduced constraints in that we did not have the option to change or alter the appearance of the device in terms of its physical human interface. For example, labels on the physical keys, the number of keys, and placement could not be changed.

One constraint that directly impacted the human interface side, was the limited methods available for showing work in progress on the actual device. This impacted our graphic designer and the author more than anyone else. To resolve this issue partially, one of the Motorola engineers wrote a simple bitmap viewing application that could be loaded onto a currently shipping Motorola PageWriter. This allowed us to take static graphics and download them to the PageWriter. The images could then be reviewed and feedback gathered.

Getting Started

The design process we followed can be illustrated in a variety of ways. The main attributes were that it was iterative, focused over time, and included a variety of ways to cross-check results. We used face-to-face visits early on to establish working relationships between the various members of the teams. This proved to be an invaluable investment.

The user profile for a typical JavaOne attendee was well documented by the conference team, through extensive surveys, and enabled us to start designing immediately. JavaOne attendees are primarily software programmers with a very small percentage of managers and marketing personnel. Attendees are mostly male (~75%), and usually have between 4 to 16 years of programming experience - very technical users.

The methodology of including users in the initial design and concept phase was not considered due to the tight schedules and complicated team communications. Instead the initial design concepts were shown to target users in the first round of user testing.

The design work on the Javelin project started with brainstorming exercises to generate fun application ideas for a personal communication device at JavaOne. Storyboards were created to document the most promising ideas. Then, a human interface prototype was created to show the basic concept of the device would operate. This phase of the design process had the added benefit of giving the teams a way to create and define common ground.

Some of the application ideas that the joint teams came up with included a conference schedule, personal conference schedule, electronic business cards, messaging, maps, and other general conference information applications. We had grand ideas about how we might use infrared and radio communication to send updates and messages over-the-air and between devices. One idea was to put the conference schedule and related information on the device and update it during the conference. Another idea was to allow a conference attendee to create a personal schedule for the conference and keep notes on the sessions they attended.

After reviewing the prototype, ideas, and proposals, the team iterated the design concepts and focused on ideas that met our user profile and schedule constraints. The application ideas that involved over-the-air and infrared were marked as requiring further feasibility testing which will be discussed in the Dreams Meet Reality section.

The prototype and storyboards were used as a basis for a common conversation with the engineers and were essential to enabling all of us to visualize how to move forward. The next stage of the project was to create a more realistic human interface prototype which would incorporate early versions of the set of applications we hoped to have on the device. The set of applications we initially focused on were the main conference schedule, personal conference schedule, messaging, business cards, maps, exhibits, show info, and device options.

Illustration 2: An early version of the Home screen using a button bar to access the applications. This is shown on a mockup of the Garnet device which was not the final device used.

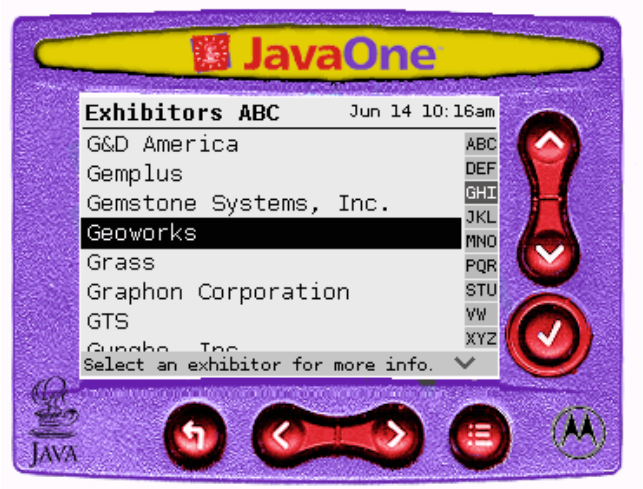

Illustration 3: An early version of the application template shown used in the Exhibits application. The template included a top and bottom bar each of which displayed a different kind of information.

Navigation: Backing up

One of the primary applications was the main conference schedule application. This application exposed many of the navigation issues the user would face when moving between different screens within an application and between applications. The question we needed to answer was what happens when the user presses the Back button on the device within the various contexts of the conference schedule application.

We started by working on the design for the Home screen from which the user would access all of the applications. This screen could always be reached in one step by pressing the Home button on the device. The initial design used a bar of buttons across the bottom of the screen. Later designs used icons to represent the applications. Illustration 2 shows an early version of the home screen with a button bar.

In parallel, we developed a template for the application screens. The template had a bar at the top of the screen which displayed a small icon, the title of the screen, and the date and time. The template had a bar at the bottom of the screen which was used to show scrolling and navigation status. An early version of the application template can be seen in Illustration 3.

Having enough of the Home screen and application template in place, we next tackled the conference schedule application. The JavaOne conference schedule was composed of a hierarchy of tracks and sessions. The conference itself lasted four days. Each day had 6 to 8 tracks that ran simultaneously. Each track was composed of a variety of sessions. Each

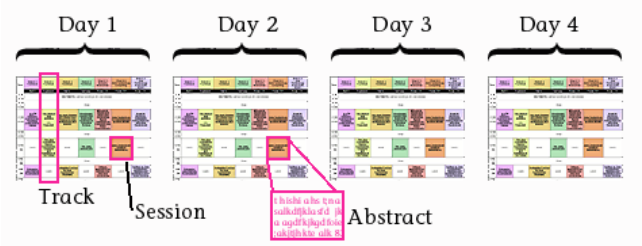

Illustration 4: The information hierarchy of the JavaOne conference schedule.

session had an abstract, speaker, room, and time associated with it. Sessions could be repeated at a later time. Illustration 4 was used to explain this hierarchy to the team.

Understanding what the Back button should do in any given circumstance meant understanding how Back worked within a single application and between applications. For example, the user could move through the conference schedule hierarchy until they found a session they were interested in. The user could then use the menu to jump to the Map application to see where the room for the session was at. The user could zoom in on the map to see more detail. What would happen if the user pressed Back at this point? Did the map unzoom? What if they pressed Back a second time? Do the user jump back to the conference session they were on? See Illustration 5.

In our first round of user testing, subjects were shown a conceptual prototype they could step through in a limited fashion - it did not have a working version of Back. Instead we queried subjects who had attended conferences before about how they typically used a conference schedule to get the information they wanted out of it. Most of the users stated that they moved around on one level as much or more than they went up or down the hierarchy. For example, a typical usage was to go to a particular day and then scan all the tracks on that day. Another common behavior was to go to a particular day and time slot and scan all the sessions going on at the time. If something interesting was seen in the

scanning of sessions, then the user would move down a level to view the abstract for that session to confirm they were really interested in it.

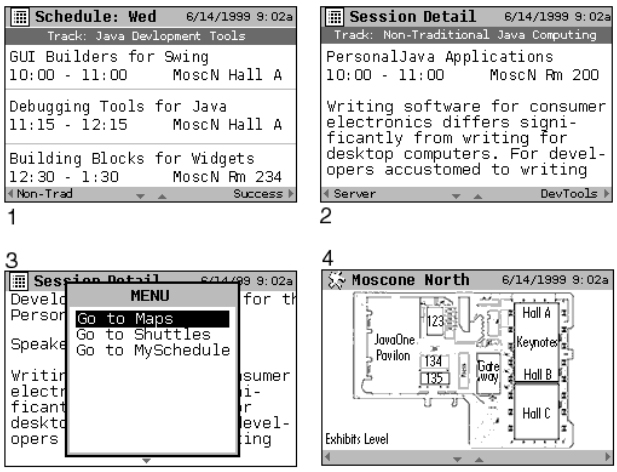

Illustration 5: The user could move from the Track screen (1) to the Session screen (2) by choosing a session. On the Session screen the user could move between sessions by traversing left/right. From the menu on the Session screen (3) the user could jump to the Map application (4).

Based on the first round of user testing, the initial proposal was that Back always went back one screen regardless of where the user was within the application. This mimicked how Back worked in web browsers on desktop systems. Given the technical profile of users we were targeting, it seemed safe to assume our users were familiar with web browsers on desktop systems.

In the second round of user testing, we implemented the Back behavior. Users could move up and down the hierarchy as well as across the hierarchy at any of the levels and each movement was saved as one Back "increment". We used this model within all the applications on the device.

What we found in the second round of user testing was that our users did not like the behavior we implemented. They were surprised by the result they got when they used Back. When the user was at the session level in the conference application, they did not want navigation across the session level captured by the Back button. They expected Back to change levels. For example, if the user had moved from Track level to Session level, then moved between several sessions, Back should return them to the Track level. Jumping to another application was treated like the user had moved to another level.

As a result of the second round of user testing, we changed the Back model to match what users wanted. Navigation on one level was not saved while navigation between levels and between applications was saved. We went though all of the applications and iterated the design to use this model. This behavior proved more usable and understandable in the long run.

Traversing About

When we started work on the Javelin project the decision about which particular Motorola device we would use had not been finalized. The proposed device at the time was a one-way pager, code-named Garnet, with a limited set of buttons. Garnet had a 240 X 160 pixels, 108 DPI, 2 bit grayscale display with up, down, left, right, back, Menu, Home, and Select buttons. (See Illustration 2.) A menu could be popped up via the Menu button. Home always took the user to the Home screen.

The Garnet employed a simple traversal model. Traversal was visible to the user via the highlight they moved around on a single screen. See Illustration 6. The highlight was the inverse video region that told the user what was going to be selected when they pressed Select. The up/down and left/right rocker buttons were used to move the highlight on the screen in any of the four directions. Items were selected via the Select button. It was very easy to figure out what to do because there were not many choices to make. It resulted in a comfortable user experience.

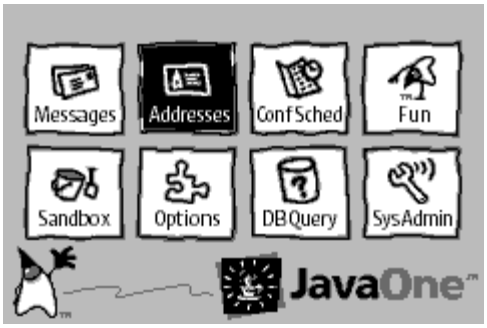

Illustration 6: The final Home screen showing showing the highlight on the Address icon. The highlight could be moved up/down or left/right using the navigation disk on the device.

We utilized this simple model in the human interface designs for the applications on the Javelin device. In early user testing, subjects appeared to understand the simple model quickly and said they liked it.

While the application team was working away, the overall management team was evaluating which device to use for the conference. The final decision was to use the new PageWriter 2000x units - not the Garnet device we thought we were targeting. This change in hardware, while not a complete surprise, did have a big impact on our designs.

The PageWriter 2000x was a two-way pager with a clamshell form factor and a very similar display to the Garnet. However, the PageWriter (see Illustration 1)

incorporated a full keyboard in addition to the limited set of keys on the Garnet. In general, given that the PageWriter key set was really a superset of the keys on the Garnet, it did not break anything in the JavaOne applications to move to the PageWriter set of keys. Home, Back, Select, and up/down, left/right navigation mapped to the PageWriter keyboard in a straightforward manner. Most of the human interface problems in changing hardware revolved around how to map the differences between the physical keys available on both devices which were used for traversing. Just because things did not break did not mean usability was not impacted.

The PageWriter 2000x used a navigation disk instead of rocker buttons to allow the user to move up/down and left/right. This navigation disk, in conjunction with the Tab key, was used to move the highlight around on a screen. The Tab key allowed the user to jump between regions on the screen. For example, the standard PageWriter human interface had a button bar at the bottom of the screen. The Tab key was used to jump between this button bar and the main application content region. The user could press the Return key or a Select key to select an item depending upon the context.

This "navigation disk plus Tab key" navigation model was a more complicated traversal model than the JavaOne applications were using on the Garnet. We evaluated whether it made sense to use this more complicated navigation model and what impact it would have to change all the application designs to support it. Given that we had already designed the applications to make use of the more simple traversal model, our conclusion was that adding the Tab key would have unnecessarily complicated the traversal model. Indeed, it would have resulted in the need for more user testing to insure we did not break things.

We had another reason to not use the Tab key. One of the keys on the Garnet was used as a Menu button that would show or hide a list of features or commands related to the application screen currently being displayed. (See Illustration 5.) We had adopted this feature from the Garnet and used it throughout our application designs. We had just completed round one of user testing and the subjects had found the Menu button concept to be useful and understandable.

We decided to see if we could use an existing key on the PageWriter for the Menu key. We considered removing the Menu key entirely but this would have resulted in major redesigns of all of the applications and the application template. It was too late in the schedule to do this. One option was to use the Tab key as the Menu key. After much consideration we adopted the Tab key as the Menu key. It was not an optimal solution, but we felt it was a workable solution given the constraints of the project.

Dreams meet Reality

At the same time at which the change in hardware was happening, the team was working towards a second user study in mid-January 1999. This study made use of the final device hardware and had a fairly complete set of applications, albeit in prototype form, that the subjects could use. We ran the study as scheduled and gained much useful information. We proceeded to finalize human interface specifications and application designs, working hard throughout the early part of February.

The joint teams had set very high expectations for what we wanted to do with the devices at JavaOne. The original plan was to hand out units to all attendees. The number of attendees was projected at 10,000 to 12,000. We wanted to update conference information and send newsbtytes over-the-air during the conference to all 12,000 units. Team members visited Moscone Convention Center to run tests with the technologies we planned to use inside the center to find out what was feasible given the expected number of attendees.

At the same time, the JavaOne personnel were ramping up the conference itself. Somewhere in mid-February we started to get updated estimates on how many attendees were expected at the conference. JavaOne has exceeded estimated attendance projections every year it has been held. This year was to be no exception. The estimates were running around 15,000 attendees with a possibility that the numbers could go as high as 20,000 which indeed they did reach.

When the feasibility information about the convention center, the new attendance figures for JavaOne, and the expected application uses were reviewed as a whole with the entire management and engineering teams, the teams had to face the reality that attempting to do over-the-air updates to 20,000 people inside of one small city block was pushing the various pieces of technology far beyond what it was currently capable of supporting. There were too many places things could fail.

Even though this was a big disappointment for the teams, the decision was made to focus our efforts on demonstrations that could be run in booths, using a transmitting tower on the exhibit floor at the conference. This decision had a huge impact on the set of demonstration applications we had been working on. The chosen set had to be completely re-evaluated based on the new focus of booth demonstrations as opposed to applications used by attendees. While this re-evaluation was going on, the clock was ticking. We had no time to start over.

From a design process viewpoint, it was critical to keep the teams moving forward and not to degenerate into chaos. The team needed something that would capture the valuable design information we had already accumulated so we could leverage and continue on with very little disruption. There were many important questions that had already been answered and no one wanted to repeat that hard work again.

A proposal was made to generate a style guide to capture design information.

Why create a style guide for what was effectively a demo project? The style guide was the best format we could think of that would capture all of the useful relevant information about the design of the human interface for Javelin and make it available to the teams as we changed direction.

The human interface style guide was based on the user studies and application designs we had already completed. This document detailed how the device operated with regard to navigation, key mapping, error messages, the visual appearance, and the behavior of the human interface components used by the applications. All of this information was used to create the new set of demo applications and the user experience on the device. See Illustration 7. The style guide was written in HTML as were all the human interface documents for the project. This enabled us to create a website that linked in all of the prior design documents with the style guide.

The author generated the style guide while the teams at both companies re-grouped and determined what the set of demonstration applications would be. The final set of applications included messaging, address book, games, over-the-air demos, and a set of options for personalizing the device.

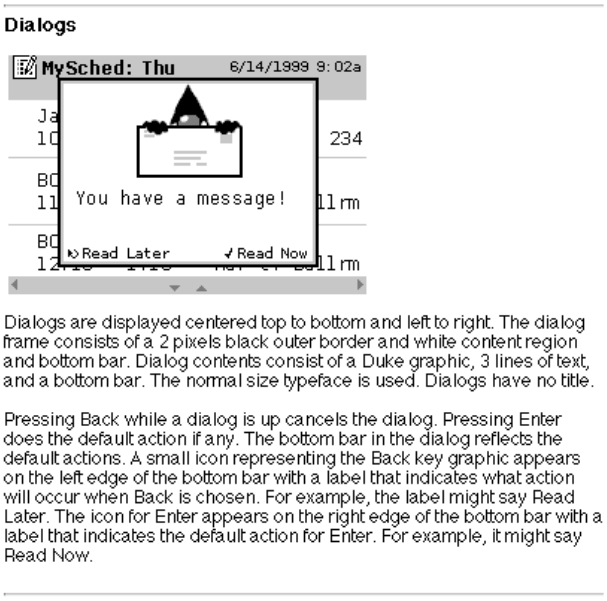

Illustration 7: A screenshot from the Javelin style guide showing the section on dialogs.

LESSONS LEARNED

There were many lessons that were learned from this project. Using the conference as a test bed and making the design process work for you are the two more significant lessons the author learned.

Conference as Test Bed

Javelin was a fun, fast-paced, not-a-product-project with a clearly defined end date. This translated into a highly constrained problem space. The advantage of this was we could take risks and try out things that would not have been appropriate for a product. The choice of what things to try out have to be weighed against the larger goals for the project which were to show off the technology.

One example of something we tried out was the graphics created for the human interface which were deliberately more creative, trendy, and cool then would have been done for a real product. See Illustration 8. People responded well to them.

Another example was the Back model which clearly would have undergone more extensive user testing before being used in a product.

While the opportunity to take some risks did not always work out, in the end, Javelin presented a coherent, easy to use, fun human interface to the various demonstration applications on the device.

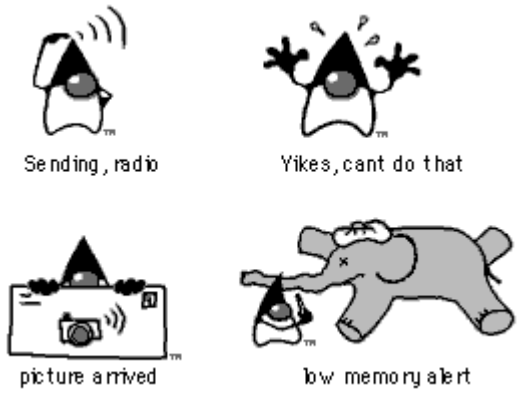

Illustration 8: These images were used in some of the alerts for Javelin. Duke is a trademark of Sun Microsystems Inc.

Making the Design Process Work for You

Knowledge of how a thing is designed and what the steps are in a design process allowed the author to adjust accordingly when the goals of the project were drastically changed. The example of how this adjustment was managed is the style guide that was created to capture the knowledge the team had created so it could be used in the next phase of work.

What had to be recognized by the designer at that point of the design process was that the project was doing a "re-start", meaning that the goals had changed and everything

was being re-evaluated. The most useful thing to do when a project does this kind of thing is to step back mentally and remind yourself of ALL the things you have learned on the project - even the things you didn't use. Then ask yourself if and how you can leverage that knowledge given the new set of goals. This will ultimately save time and energy. The best way the author has found to leverage prior work has been to capture it in a format that can be used by the entire team. A style guide worked well in this project. A prototype might work better in a different project.

ACKNOWLEDGMENTS

The author would like to thank all of the Javelin team members from Motorola, Inc. and Sun Microsystems, Inc., the JavaOne Sun people, the members of the User Experience Group at Sun in the Consumer Embedded division, and Kate Withey, Withey Design, for all their support and help in making this project the success it was. Special thanks to Dana Miller for user testing support and Hugh Johnson for be willing to answer all my questions.

REFERENCES

1. The Consumer Products User Experience Style Guide, Sun Microsystems Inc., Palo Alto, CA, USA, unpublished manuscript.
2. Motorola PageWriter 2000 User's Manual. Flex Programmable Products and Applications Division, Boynton, Florida, USA. 1997, 1998.
3. Bergman, Eric. *Information Appliances and Beyond: Interaction Design for Consumer Products*, Morgan Kaufman Publishers, San Francisco, CA, USA, December 1999, ISBN:1-55860-600-9