Generating mobile device user interfaces for diagram-based modelling tools

Dejin Zhao[1], John Grundy[1, 2] and John Hosking[1]

[1]Department of Computer Science and [2]Department of Electrical and Computer Engineering,
University of Auckland, New Zealand

{john-g, john}@cs.auckland.ac.nz

Abstract

Mobile display devices such as phones and PDAs have become very widely available and used. However, most content on these devices is limited to text, static images and motion video. Displaying and interacting with dynamic diagrammatic content on such devices is difficult, as is engineering applications to implement such functionality. We describe a set of plug-in components for a meta-diagramming tool that enable a diagram type to be visualized and interacted with on mobile devices. Key features of our approach include generating diagram content from an existing meta-tool, run-time user configuration of diagram appearance and navigation, and multi-level, zoomable diagrams and diagram content. We describe our experiences prototyping, using and evaluating this new mobile device diagramming technology.

Keywords: mobile user interfaces, diagrams on mobile phones, collaborative design

1 Introduction

Mobile devices such as PDAs and phones have become very widely available and used in recent years. Many offer a range of sophisticated content including styled text, rich graphical images and full-motion video streaming. Many now offer quite high resolution display capabilities along with a variety of interaction techniques, including buttons and stylus-based sketching. However, very few applications for mobile devices provide dynamic diagrammatic display and almost none interactive diagram editing.

Diagrammatic applications include design tools for a variety of domains such as information structuring and browsing, concept sketching and discussion, project management, and software and user interface design. Examples of diagram types include trees for file and concept hierarchies, graphs for network, relationship and dependency specification, and various domain-specific notations, such as Gantt and Pert charts for project management, UML diagrams for software design, and various visual languages for concept analysis and programming. Traditional diagramming tools for such applications are built using thick client, window-based interfaces. This limits them to desktop and laptop computers. To improve access to these tools, thin client diagramming tools [1, 10] have been proposed and prototyped, typically using a web browser for viewing and sometimes editing diagrams. The advantages of this approach include no need to install software on each user's computer and use of web-based architectures and interaction to build and access the diagramming tools. However, unlike desktop PCs and laptops, mobile devices have many constraints on screen display size, interaction techniques and bandwidth over mobile cellular networks. This makes existing thick- and thin-client diagramming approaches unsuitable for these devices.

We wanted to explore the issues involved in making thin-client diagramming applications available on mobile devices like PDAs and mobile phones. To this end we have built a proof-of-concept extension to a meta-CASE tool, Pounamu, allowing each diagramming tool specified in Pounamu to be effectively realized on mobile devices. We begin by presenting our motivation and surveying related research. Then we give a detailed walkthrough of the architecture and our approaches with illustration of use cases. We briefly describe our implementation based on Nokia's MUPE framework [19] before concluding with a summary and future work.

2 Motivation

We have been developing a meta-CASE tool called Pounamu [24]. Pounamu is used to interactively define new domain-specific tools and to provide a framework for realizing these tools. Some extensions have been developed to extend access to tools and support collaborative editing. These include a thin client diagramming plug-in [1] and a collaborative work plug-in [15]. These extensions of Pounamu provide different kinds of user interfaces for diagram modeling tools built with Pounamu, including single user thick client, multi-user thin client, and multi-user collaborative work support.

Figure 1 (a) shows an example diagramming tool generated by Pounamu, a UML CASE Tool, in use. Another tool, a project management tool, is shown in Figure 1 (b). Figure 1 (c) shows a web client user interface implemented by Pounamu/Thin, our plug-in extension to support web-based diagramming for Pounamu tools. While these modelling tools provide a range of diagramming features, they are constrained to use on desk-top or laptop PCs.

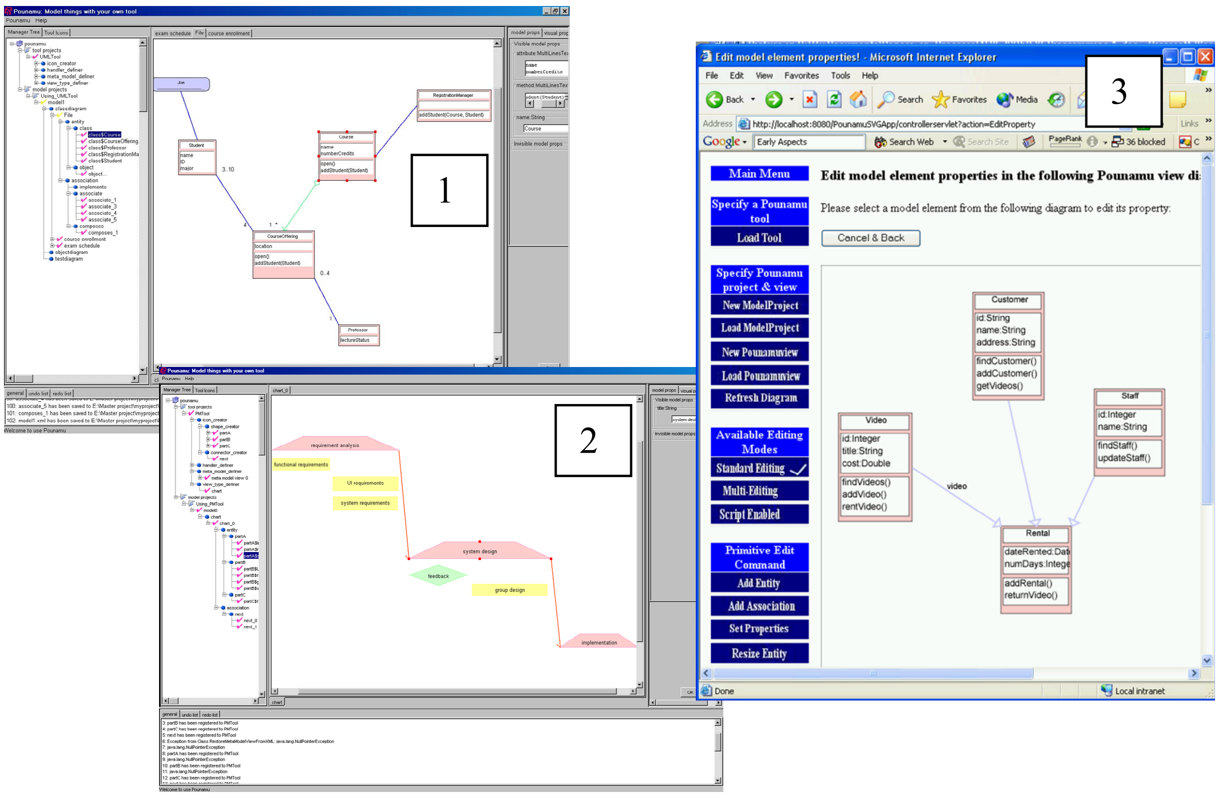


Figure 1. (a) UML tool in Pounamu; (2) Project Management tool in Pounamu; (3) Pounamu/Thin interface.

As discussed above, there are many applications for diagram-based user interfaces that users could potentially want deployed on mobile devices. To support such pervasive access and mobile collaborative model design, we wanted to make Pounamu modelling tools accessible on a range of mobile devices, such as mobile phones and PDAs. We also wanted to explore the technical and usability issues associated with building complex diagram-based user interfaces on mobile devices. Our approach to achieving this was to adopt a similar mechanism to our web-based Pounamu/Thin plug-in. We built a new server using Pounamu's plug-in mechanism (Pounamu/Mobile). This supports multi-user access to tools generated by Pounamu, through thin clients on disparate mobile devices.

Mobile devices are however constrained in a number of ways over desktop clients making them particularly challenging on which to realize diagram-based interfaces. Some of these constraints include:

- The relatively low processor speed, small memory and storage of most mobile devices
- The wide range of mobile devices and the issue of deploying applications onto devices running different operating systems
- Diagram rendering limitations of most current mobile devices, particularly for diagrams originally defined for thick-client PC display and editing.
- Techniques to display large diagrams on small screens and yet still keep the diagram meaningful
- Techniques to navigate large diagrams and between multiple views (diagrams) of a model
- Supporting user preferences so that different users can specify different diagram content rendering, zooming and navigation configurations via their mobile device

3 Related Work

A number of systems have been developed to assist in developing mobile and thin-client user interfaces. Several provide overviews of complex information such as web pages, on mobile devices, using e.g zoomed-out copies of pages [9, 19] or summarized versions of the page [1, 7]. Generally these “content outlining” approaches transform pages into sets of tiles [19, 5, 1] and while they assist navigation they are not suitable for most diagrammatic representations. Some mobile browsers combine tiling with zooming, so that users can access individual tiles from an overview [9, 7, 8]. These increase the amount of complex content that can be reasonably accessed and browsed on a small screen mobile phone. Approaches to collapsing page content allows users to zoom into relevant areas by collapsing areas that are less relevant [1] and also increase the ability of mobile users to access complex information that otherwise could not be presented on a single small screen.

A number of mobile applications have been developed that provide diagrammatic representations of complex information. These include tourism and travel planning, map usage and management [12, 16, 21, 22, 24]. Most of these applications have read-only diagrams that do not support editing. A number support some form of zoom-based display enabling users to focus in on areas of interest quickly and navigate complex information spaces. However many of the panning and zooming capabilities are limited to either the mobile device's standard characteristics or single points of zoom and focus. Our experiences have indicated that multiple points of focus are desired by users when working with the kinds of diagrammatic representations illustrated in the previous section.

Thin-client, web browser-based approaches have become popular for various diagramming applications [1, 10, 9, 14]. At present most such systems focus on supporting standard web browsers on desktop or laptop PCs, rather than smaller screen mobile PDA or phone devices. However, many of these web browser-based diagramming tools use architectures that could support mobile device-based diagrammatic rendering and editing.

A number of architectures have been developed to support multi-device user interface construction [20, 3, 11, 15, 23]. These systems aim to provide a single way of specifying user interfaces that may be presented on a wide range of devices. To date however most of these interfaces have been limited to text and form-based data rather than richer diagrammatic content. Some use an approach of translation from an abstract XML format into a device-specific format. However most do not have a generic diagramming application as their "application server", instead implementing custom diagramming support specifically for the mobile device. While this allows some optimisations to be made in terms of content production, it has the disadvantage of not being able to leverage much of the work on diagramming meta-tools and architectures.

4 Our Approach

In this section, we briefly explain the architecture of our Pounamu/Mobile thin-client diagramming extension and our approaches to addressing the above requirements. The following section illustrates how our extension works using two prototype design tools, a UML design tool and a Project Management tool.

Figure 2 shows a high-level architecture of our Pounamu/Mobile approach. The Pounamu meta-tool runs on a host and behaves as the application server/database manager. It supplies application data to the Pounamu MUPE server, including tool specifications, models, and multiple views (diagrams) of models. The Pounamu MUPE server generates user interfaces for diagramming tools according to this data and user configurations, which is explained later in this section. Mobile hosts can have differing screen size, display and input capabilities, and differing connectivity to the Pounamu MUPE server (Bluetooth, WLAN, GPRS etc).

We addressed the requirement for deployment of Pounamu/Mobile interfaces on various devices by using Nokia's MUPE mobile client/server technology. MUPE is a mobile thin client technology developed by Nokia consisting of a MUPE server and MUPE client browser. The MUPE server responds to client requests by generating and sending back pages in XML, which can be browsed in a MUPE client. MUPE clients are based on Java MIDP, which is enabled on most newer mobile devices. By using MUPE, diagrammatic user interfaces generated by our system can be easily deployed on a wide range of MIDP-compatible devices. The server sends XML-encoded data to the MUPE handset client using a protocol optimised for the MUPE client. MUPE XML markup includes GUI widgets like text, bitmaps, input fields and so on, along with canvas-based drawing and client-side scripting for highly interactive interfaces. The MUPE client sends XML-encoded requests back to the MUPE server for processing. MUPE differs from other mobile client-side applications such as pure MIDP2.0 by trying to provide a higher-level markup language for complex interface specification along with a client-server request/response protocol in XML.

Our Pounamu/Mobile server is implemented as a set of components hosted on a MUPE-supporting server. These components include diagram view rendering and manipulation, diagram shape and connector property editing facilities, and multi-user tool configuring facilities. Any MIDP2.0 supporting client device can have the MUPE client application installed and thus render Pounamu/Mobile diagrams and provide editing support.

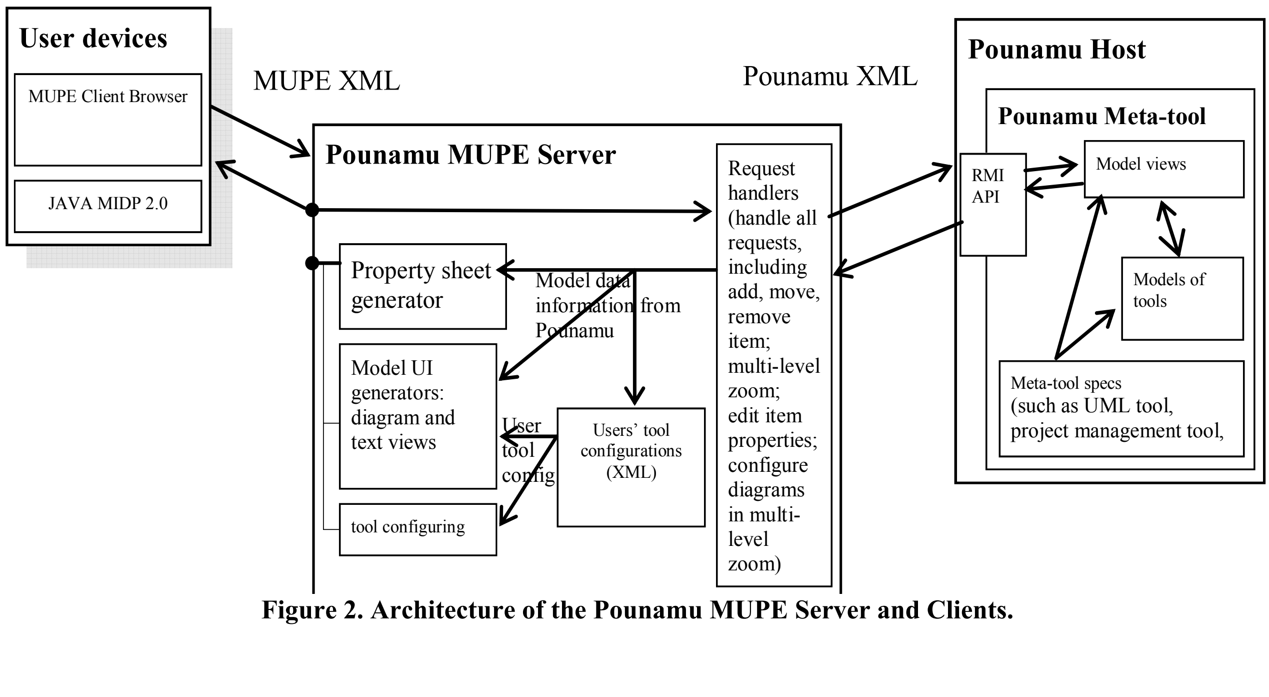

User devices
MUPE Client Browser
JAVA MIDP 2.0
MUPE XML
Pounamu XML
Pounamu MUPE Server
Property sheet generator
Model data information from Pounamu
Model UI generators: diagram and text views
User tool config
Users' tool configurations (XML)
tool configuring
Request handlers (handle all requests, including add, move, remove item; multi-level zoom; edit item properties; configure diagrams in multi-level zoom)
Pounamu Host
Pounamu Meta-tool
RMI API
Model views
Models of tools
Meta-tool specs (such as UML tool, project management tool,

Figure 2. Architecture of the Pounamu MUPE Server and Clients.

To help address the rendering and resolution limitations of most mobile devices, we developed a multi-level rendering and zooming capability. This facility allows users to define multiple representations for a single diagram at different levels of detail. While manipulating model views, each diagram element can be separately selected and zoomed among multiple levels. Automatic zooming of elements is supported as users navigate a large view.

We use several navigation approaches to improve user interaction with diagramming tools on mobile devices. Button-panning let users quickly move around in a big diagram using mobile phone buttons. A floating zooming window allows users to easily and quickly zoom to interesting areas from an overview of a large diagram. Pounamu/Mobile supports user preferences by using a multi-user runtime application configuration facility. This allows users to specify different diagram content, zoom, and navigation configurations via their mobile device. User configurations are stored in XML format in the Pounamu/Mobile Server. At run-time the system obtains model information to display from the Pounamu application server and generates diagram views according to each user's configuration. Default user configurations are generated and these may be tailored to enhance users' display and interaction.

5 Example Usage

In this section we illustrate example mobile user interfaces generated by our Pounamu/Mobile from two quite different prototype tools. One is a UML CASE tool and the other is a Project Management tool, as shown in thick client form in Figure 1. Our Pounamu/Mobile components are completely tool-independent hence no code changes are necessary to support rendering and editing of *any* diagramming tool specified in our original Pounamu meta-tool.

Figure 3 (a) shows the initial Pounamu/Mobile screen after logon, showing a list of viewing and editing options. After selecting a diagram to display, an initial overview of it is generated. Two overviews of selected views using the two tools are shown on two different mobile phones in (b) and (c). An overview of the same model view may appear differently on different mobile devices. This is because the overview is generated by proportionally shrinking original diagrams to fill them onto the small screens of mobile devices. The proportion is calculated according to the size of the selected Pounamu view to display and the screen size of the client device requesting a page. Detailed contents of diagrams are filtered out in overview mode because they are thumbnails and hard to see. Users can press direction keys to select diagram shapes and connectors. The status bar underneath displays certain details of the current selected item, such as the name and type of the selected item. Figure 3 (b) shows one class diagram view on a small colour phone. Figure 3 (c) shows a Gantt chart view from a project management model on the Nokia 7650.

Figure 4 shows an example of multi-level zooming and pan navigation in Pounamu/Mobile views. To zoom in/out diagram shapes to various detail levels, firstly a user selects a diagram shape from the model view, and then presses a pre-defined hot key to zoom into a particular level.

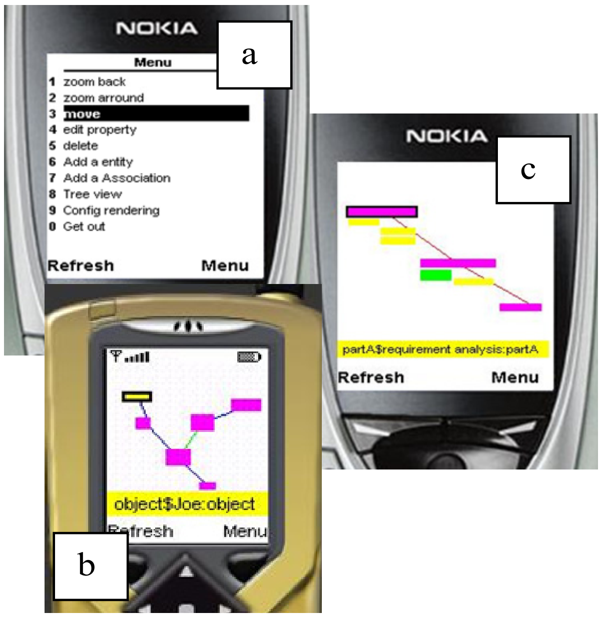


Figure 3. (a) main menu of Pounamu/Mobile tools; (b) overview of class diagram view on a small colour phone; (c) overview of a Gantt-chart view on Nokia 7650.

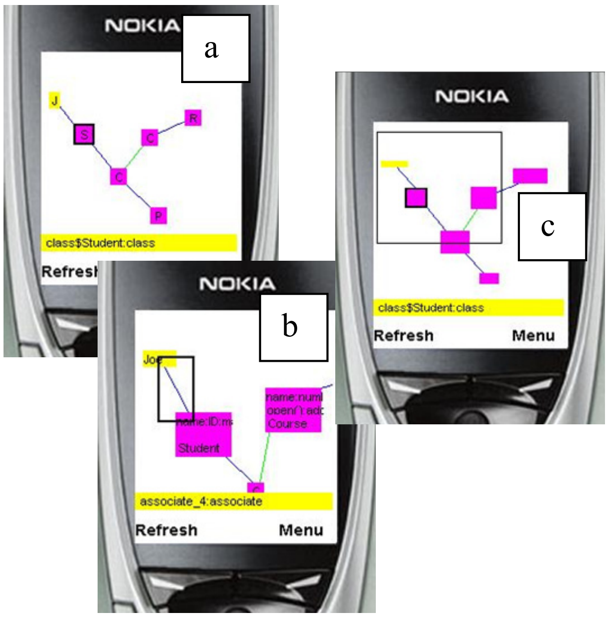


Figure 4. (a) the class diagram view with all diagrams zoomed in level 1; (b) same class diagram view with diagrams zoomed to different levels; (c) pan navigation to show the current position and quick zoom in to the selected area.

In Figure 4 (a), all diagram elements are at "zoom level 1", providing a high level overview of the diagram but no details for each element. In Figure 4 (b), one object entity is zoomed in level 2, and two class entities are zoomed in level 3. The user can select individual elements and zoom them in or out, or can enable an auto-zoom facility that magnifies the selected item and its immediate neighbours while de-magnifying those further away. This forms a basic distortion-oriented display. As each Pounamu diagram type can have different element types and connectivity, different view types have different auto-zoom behaviours.

To use pan navigation, a user selects pan navigation from the options menu or a hot-key button. They can zoom out to the top-level overview and a floating window indicates the current zooming area in the model view, as shown in Figure 4 (c). The floating window can be moved across the diagram using mobile phone keys and selecting a hot-key magnifies elements in the selected zooming area.

Figure 5 shows another example of multi-level zooming facility this time with the Gantt chart model view. The overview shows the Gantt chart elements at their lowest zoom level, as in Figure 5 (a), giving an indication of the overall flow of project work only. Elements can be magnified to see their relative duration and details of the task they indicate, as in Figure 5 (b). Users can navigate around the task flow using the pan facility and auto-zoom. For example, as the user moves the selection focus down the tasks in the Gantt chart, de-selected tasks are zoomed to a lower level, as shown in Figure 5 (c).

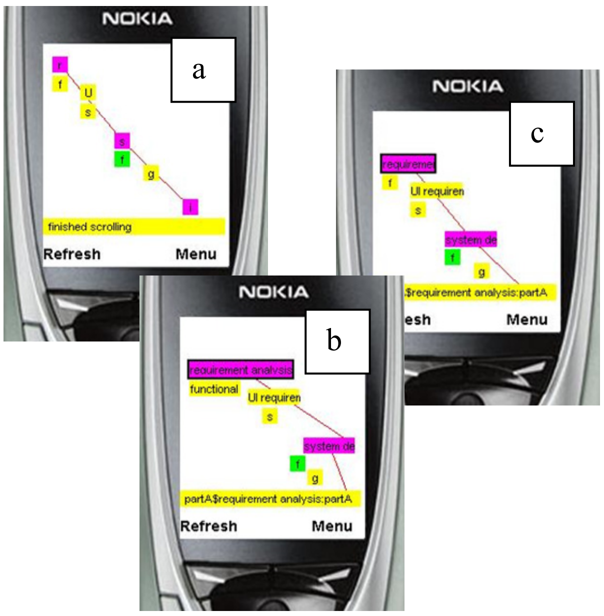
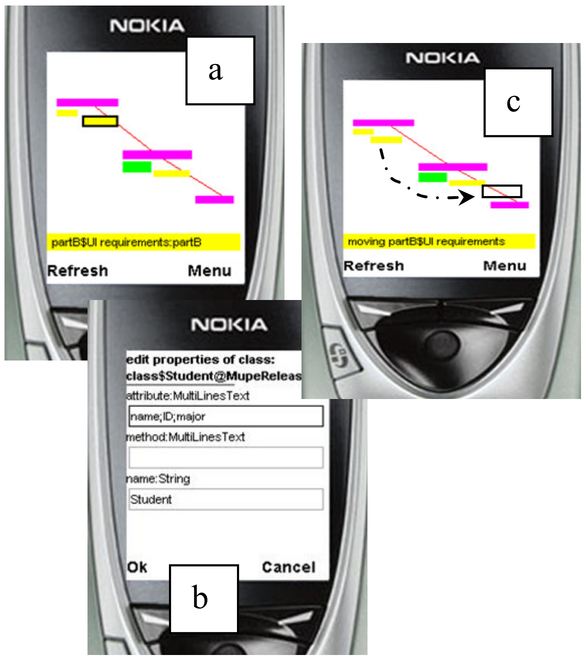


Figure 5. Multi-level zooming with the Gantt-chart view of a project management model.

Adding, moving, deleting, and editing items in a diagram require similar interactions through the generated Pounamu/Mobile User interfaces. Firstly a user selects an item to manipulate, and then chooses a menu command or a hot-key indicating the action they want to perform on that item. For example, to move a shape, the user selects a shape then repositions the shape using the mobile device's direction keys. They then confirm the new position by pressing e.g. the Select hot-key on the device. Items are added to a place on the screen by moving the pan selection floating window and narrowing it down to an unoccupied space with hot-keys.

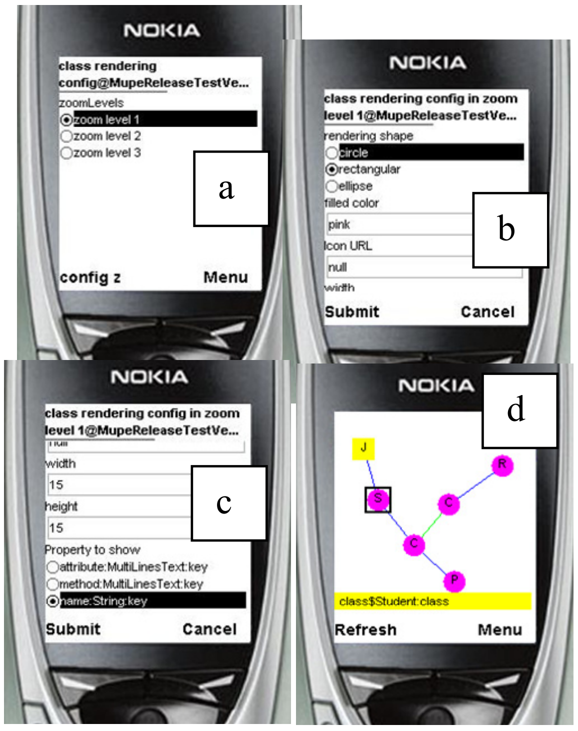
To edit properties of an item, the user selects the item, as shown in Figure 6 (a), then selects a hot-key or menu option to edit its properties. A list of the element's properties is displayed which the user edits with conventional mobile device interactions, as shown in Figure 6 (b). The user then selects the OK menu to submit the values back to server. After processing the user request, the server sends back an updated view.



edit properties of class:
class$Student@MupeReleas
attribute:MultiLinesText
name;ID;major
method:MultiLinesText

name:String
Student

Figure 6. (a) select an element; (b) edit item properties; (c) moving an element.



class rendering config@MupeReleaseTestVe...
zoomLevels
[✓ zoom level 1]
zoom level 2
zoom level 3

class rendering config in zoom level 1@MupeReleaseTestVe...
rendering shape
circle
[✓ rectangular]
ellipse
filled color
pink
Icon URL
null
width

class rendering config in zoom level 1@MupeReleaseTestVe...
width
15
height
15
Property to show
attribute:MultiLinesText:key
method:MultiLinesText:key
[✓ name:String:key]

Figure 7. (a-c) user configuring class diagram representations in multi-level; (d) the class diagram with all items zoomed to level 1 as specified by users.

One issue we repeatedly encountered when building multi-device user interfaces is the inability of generated interfaces to suit all potential users. In particular, when providing multi-level zooming capability on a diagram, we found different users want to have different element renderings for different zoom levels. Hence we have built individual user configuration capability into Pounamu/Mobile. Figure 7 (a) shows an example of tool configuration via mobile user interfaces at runtime to specify different element appearance at different zoom levels.

To configure diagram representations, a user selects the type of entities that they want to specify the appearance of and then select the menu "Config rendering". The mobile server finds the user's Pounamu tool configuration in xml format and generates configuration user interfaces. Then the user selects the zoom level that they want to specify the representation for. A diagram representation currently can be specified by using color, size, icon shape, and properties to show. Figure 7 (b) shows the result of a user configuring a UML class diagram view in which all diagrams are zoomed in level 1. Diagram representations are different from the one shown in Figure 5 (a) as different shapes have been selected for the icon representations.

6 Design and Implementation

The Pounamu/Mobile server is made up of a set of MUPE server components that interact with the Pounamu meta-tool to synthesise user interfaces. A single instance of the Pounamu meta-tool acts as both the meta-tool capability, allowing definition and modification of diagramming tools, and the data and processing application server for Pounamu/Mobile. All Pounamu/Mobile interactions are handled in a similar way.

Our Pounamu/Mobile server is mainly composed of 3 components: Request Handling, UI Generation, and User Tool Configuration. Request handlers respond to all requests from MUPE clients. They hold the functional services supplied by the server. The Request Handling component is responsible for handling user requests, communicating with Pounamu application, and notifying corresponding UI generators to generate pages. UI generators generate user interfaces of Pounamu views, property forms, and tool configurations.

Each UI generator contains a set of objects, which have their own visual representation templates in an XML format. Each registered user has his/her own tool configuration as an XML file for each tool he/she has loaded. These configuration files are stored in the Pounamu/Mobile Server. To generate a diagram view, the Model UI generator collects and combines information from multiple sources, including model view information from the Pounamu application server, user personalized specification of diagrams from user's tool configuration file, screen size of client device, and diagram rendering templates of rendering objects.

Figure 8 shows an example set of interactions when editing a diagram in a Pounamu/Mobile client. If the user asks for an element to be deleted, a MUPE XML event message is formulated by the MUPE client on the device and sent to the MUPE server (1). A Pounamu/Mobile request handler determines the diagram update required (2) and formulates a Pounamu API call to action the update on the diagram (3). The update results in the element being deleted from the Pounamu view (4), deleting the model element and impacting other views of this deleted element (5). Once the Pounamu server acknowledges that the update has successfully occurred, Pounamu/Mobile re-reads the updated view's Pounamu XML from the Pounamu server (6). It then synthesises a new view in the MUPE XML format, using the user preferences and Pounamu view XML to do this (7). The MUPE client is returned a new page to display, the updated diagram content (8). Various optimisations are possible, including caching the MUPE XML in the MUPE server and only updating affected portions. However we have found response time when re-generating the whole diagram content is adequate and this simplifies the process considerably.

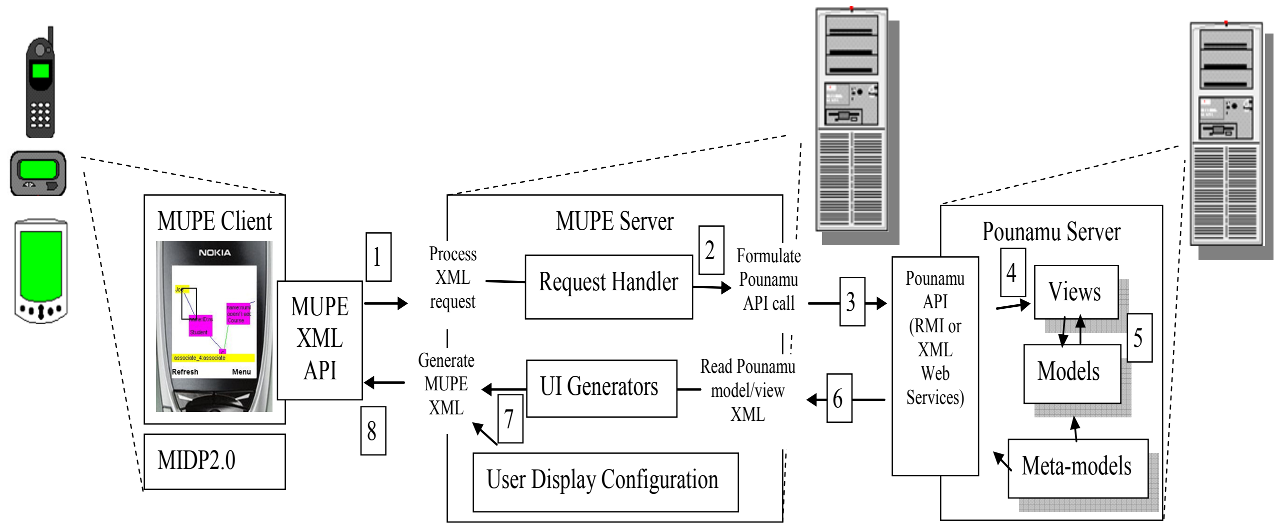


Figure 8. Basic component interactions in Pounamu/Mobile.

7 Discussion

We have tested our Pounamu/Mobile thin client plug-in components using several visual modelling tools specified by Pounamu. They include a UML CASE tool with class, collaboration, use case, sequence diagram and deployment diagram views, a project management tool with Gantt, pert chart and work breakdown views, and a business strategy modelling tool with process modelling views. For each no code change to Pounamu/Mobile was required and only a new configuration specification by the end user for each view type if s/he wanted to modify the default configuration.

We have evaluated the usability and performance of these Pounamu/Mobile tools with several experienced mobile phone and PDA users. We carried out experiments with the MUPE and Pounamu servers installed on a workstation and users accessing the MUPE mobile interface via the MUPE client simulator, running on separate workstations in different locations. We used two basic experiments, one using the project management tool prototype with users carrying out basic task planning and co-ordination activities. The second involved users reviewing and making minor modifications to (fairly simple) UML class diagrams representing a data model for a business application. Users carried out both concurrent (synchronous) activities as well as asynchronous activities in both experiments. Users also used the Pounamu desktop application to carry out these same tasks as a comparison. We used the exact same tool specifications from Pounamu for both Pounamu/Mobile experiments as for the desktop editing experiments.

Results from these evaluations indicate that overall our Pounamu/Mobile approach provides similar viewing and editing support to the capabilities of our previous desk-top thick client and web-based thin client versions of Pounamu. Users were able to access both navigation and editing facilities in the mobile device but as expected the editing of diagrams was much more difficult than desktop or even web browser-based interfaces. Navigation issues on a small screen were to some degree mitigated by Pounamu/Mobile's multi-level zoom capability and by users being able to specify multiple visual representations for a single diagram shape. Users found these features to be almost essential in order to support complex diagram browsing and drill-down to detailed item viewing and editing.

Users indicated that more automated support by the Pounamu tool would help diagram editing e.g. automated placement, layout and auto-zooming both during browsing and editing. This was particularly so for the UML tool which provides little of such support. This finding is in contrast to when using these tools through the thick-client Pounamu interface where automatic layout by the tool is not preferred by users. The difference with the mobile application interface appears due to the more limited placement control users have with non-stylus mobile devices we evaluated. The Gantt chart tool does provide some automatic layout support which users did find helpful to have on the mobile device. In both examples as soon as diagrams get moderately large (greater than about 15 items) they became very difficult to use on the mobile interface. Overall users thought the approach to be effective for making complex diagrammatic content accessible via a mobile device and allowing other users to simultaneously access desktop and web browser versions of the content. A number of usability enhancements to make browsing and editing faster and easier are required to make the approach feasible for real work. These include user configurable hot-keys to speed up some functions, especially browsing, and user configurable layout in addition to zoom shape specification. In addition, support for collaborative interactions needs to be improved, as has been done in the work of others [13]. The element-by-element zooming and auto-zoom capabilities of Pounamu/Mobile were thought to be potentially useful in the desktop Pounamu editing tools by several users.

Like other researchers we have encountered some hard technical challenges when developing Pounamu/Mobile. The graphic rendering limitations of MIDP and difficulty of user interaction on mobile phones means that diagrams specified and rendered on mobile devices can not be as complex as on a PC. However, as mobile devices continue to become more powerful we expect to be able to achieve more complex diagramming rendering on mobile devices. Icon specification is currently quite cumbersome using the MUPE mobile device interface. This would be much easier and flexible on pen-based PDAs than on the key-based phones that we have used in our experiments to date. Because of the memory limitations on current mobile devices, very large diagrams can't be rendered, even when using MUPE thin client technology. Instead, at present, smaller overlapping Pounamu views must be constructed. Again the rapid evolution of mobile hardware will remove this limitation. We currently use a basic layout algorithm to transform Pounamu views into an overview for mobile devices. This works fine for e.g. our UML tool which is not so sensitive to spatial layout. However, some tools use layout as an important component of the diagram such as Gantt charts in our project management tool.

Key areas for future work we are planning include extending our work to cover more types of mobile devices with other styles of input, such as stylus input on PDA and Tablet PC and speech input. As we discussed above, specifying representations of diagrams will definitely be improved by using stylus input. We would also like to apply our approaches to mobile technologies other than MUPE, such as Mobile SVB, Symbian development framework in C++, and Microsoft .Net Mobile version. We want to improve the layout algorithm used for overview diagrams to better produce layouts for those diagram types relying heavily on spatial relationships. Potential approaches may include spatial reasoning and other related approaches on mobile devices [24]. We would also like to integrate some of our previous work on collaborative editing support [15] to the Pounamu/Mobile user interfaces. Back-porting the zooming feature of Pounamu/Mobile to the desktop Pounamu editing tools appears to be useful.

8 Summary

We have built a prototype mobile thin client plug-in component for a meta-CASE tool, Pounamu. This means that ANY Pounamu-specified visual designing tool can be accessed via thick client, web-based thin client, and mobile thin client interfaces without additional programming. Our main contributions include the following: (1) our proof-concept work demonstrates that complex diagram based user interfaces specified for thick client diagramming tools can be supported on mobile devices using the same specification; (2) we have proposed a set of approaches for manipulating complex diagrammatic models using small screen interfaces, including multi-user runtime configuration and multi-level zooming; (3) three exemplar tool applications have been tested and evaluated using the mobile thin plug-in demonstrating efficacy of the approach.

References

1. Baudisch1, P., Xie, X., Wang C., and Ma, W.Y. (2004): Collapse-to-Zoom: Viewing Web Pages on Small Screen Devices by Interactively Removing Irrelevant Content. In Proc.UIST '04.
2. Björk, S., Holmquist, L.E., Redström, J., Bretan, I., Danielsson, R., Karlgren, J., and Franzén, K. (1999): WEST:: a web browser for small terminals. In *Proc. UIST'99*, pp. 187–196.
3. Bonifati, A., Ceri, S., Fraternali, P., Maurino, A. (2000) : Building multi-device, content-centric applications using WebML and the W3I3 Tool Suite, Proc. Conceptual Modelling for E-Business and the Web, LNCS 1921, pp. 64-75.
4. Buyukkokten, O., Garcia-Molina, H., Paepcke, A. (2001): Text Summarization for Web Browsing on Handheld Devices. *Trans. Inf. Sys. 20* (1):82-115.
5. Buyukkoten, O., Garcia-Molina, H., Paepcke, A., and Winograd, T. (2000): Power browser: efficient web browsing for PDAs. In *Proc. CHI'00*, ACM Press, April 1-6, 2000, The Hague, pp. 430–437.
6. Cao, S., Grundy, J.C., Hosking, J.G, Stoeckle, H., Tempero, E. (2004): An architecture for generating web-based, thin-client diagramming tools, 2004 IEEE Int. Conf. on Automated Software Engineering, Linz, Austria, Sept 25-29 2004, IEEE.
7. Chen, Y., Ma, W.Y., and Zhang, H.J. (2003): Detecting Webpage Structure for Adaptive Viewing on Small Form Factor Devices. In *Proc. WWW'03*, 20-24 May 2003 Budapest, Hungary, pp 225–233.
8. Chen, L.Q., Xie, X., Ma, W.Y., Zhang, H.J., Zhou H.Q., Feng H.Q., (2002): DRESS: A Slicing Tree Based Web Representation for Various Display Sizes. *Microsoft Research MSR-TR-2002-126*.
9. Eisenstein, J. and Puerta, A. (2000): Adaptation in automated user-interface design, Proc. 2000 Conference on Intelligent User Interfaces, New Orleans, 9-12 January 2000, ACM Press, pp. 74-81.
10. Gordon, D., Biddle, R., Noble, J. and Tempero, E. (2003); A technology for lightweight web-based visual applications, Proc. 2003 IEEE Conf. on HCC, Auckland, NZ, 28-31.
11. Grundy, J.C. and Zhou, W. (2003): Building multi-device, adaptive thin-client web user interfaces with Extended Java Server Pages, In Cross-platform and Multi-device User Interfaces, Wiley, 2003.
12. Luz, S., and Masoodian, M. (2004): A Mobile System for Supporting Non-Linear Access to Time-Based. In Proc. 7th International Working Conference on Advanced Visual Interfaces, Gallipoli, Italy, 25-28 May, ACM Press, 454-457.
13. Luz, S., Masoodian, M., and Weng, G. (2003): Browsing and Visualisation of Recorded Collaborative Meetings, In Proc. 10th International Conference on Human-Computer Interaction, Crete, Greece, 22-27 June, vol. 2, 148-152.
14. Mackay, D., Biddle, R. and Noble, J. (2003): A lightweight web based case tool for UML class diagrams, In Proc. 4th Australasian User Interface Conference, 2003.
15. Marsic, I. (2001): An architecture for heterogeneous groupware applications, Proc. International Conference on Software Engineering, May 2001, IEEE CS Press, pp. 475-484.
16. Masoodian, M., and Budd, D. (2004): Visualization of Travel Itinerary Information on PDAs. Conference Proceedings of AUIC 2004, Proc. 5th Australasian User Interface Conference, Dunedin, New Zealand, 18-22 January, pp. 65-71.
17. Mehra, A. Grundy, J.C. and Hosking, J.G. (2004): Supporting Collaborative Software Design with a Plug-in, Web Services-based Architecture, Proc. ICSE 2004 Workshop on Directions in Software Engineering Environments, May 25 2004, Edingurgh, Scotland, IEE Press.
18. Milic-Frayling, N. and Sommerer, R. (2002): SmartView: Enhanced Document Viewer for Mobile Devices. *Microsoft Research Technical Report MSR-TR-2002-114*.
19. MUPE, www.mupe.net
20. Palm Corp. (2001): Web Clipping services, www.palm.com, 2001.
21. Rossel M. (1999) : Adaptive support: the Intelligent Tour Guide. 1999 Int. Conf. Intelligent User Interfaces. ACM. 1999, New York, NY, USA.
22. Stephanidis, C. (2001): Concept of Unified User Interfaces, In User Interfaces for All - Concepts, Methods and Tools, Laurence Erlbaum Associates, August 2001, pp. 371-388.
23. Van der Donckt, J., Limbourg, Q., Florins, M., Oger, F., and Macq, B. (2001); Synchronised, model-based design of multiple user interfaces, Proc. 2001 Workshop on Multiple User Interfaces over Internet.
24. Wobbrock, J., Forlizzi, J., Hudson, S., Myers, B. (2002): WebThumb: interaction techniques for small-screen browsers. In *Proc.UIST '02,* pp. 205–208.
25. Zarikas, V., Papatzanis, G., and Stephanidis, C. (2001): An architecture for a self-adapting information system for tourists, 2001 Workshop on Multiple User Interfaces over the Internet.
26. Zhu, N., Grundy, J.C. and Hosking, J.G. (2004): Pounamu: a meta-tool for multi-view visual language environment construction. Proc. 2003 IEEE Int. Conf. on Human-Centric Computing, Rome, Italy, 2004, IEEE.