Collaborative End-User Development on Handheld Devices

Navid Ahmadi
Faculty of Informatics
University of Lugano
Lugano 6904, Switzerlands
navid.ahmadi@lu.unisi.ch

Alexander Repenning
AgentSheets Inc.
6560 Gunpark Dr. Suite D
Boulder, CO 80301
alexander@agentsheets.com

Andri Ioannidou
AgentSheets Inc.
6560 Gunpark Dr. Suite
DBoulder, CO 80301
andri@agentsheets.com

Abstract

Web 2.0 has enabled end users to collaborate through their own developed artifacts, moving on from text (e.g., Wikipedia, Blogs) to images (e.g., Flickr) and movies (e.g., YouTube), changing end-user's role from consumer to producer. But still there is no support for collaboration through interactive end-user developed artifacts, especially for emerging handheld devices, which are the next collaborative platform. Featuring fast always-on networks, Web browsers that are as powerful as their desktop counterparts, and innovative user interfaces, the newest generation of handheld devices can run highly interactive content as Web applications. We have created Ristretto ^{Mobile}, a Web-compliant framework for running end-user developed applications on handheld devices. The Web-based Ristretto ^{Mobile} includes compiler and runtime components to turn end-user applications into Web applications that can run on compatible handheld devices, including the Apple iPhone and Nokia N800. Our paper reports on the technological and cognitive challenges in creating interactive content that runs efficiently and is user accessible on handheld devices.

1. Introduction

The end-user's role in software production has evolved over the years. Initially, end-users were only using software without the ability to modify anything. Later, end users were enabled to customize software. More recently, during the last decade, especially with the emergence of the Web, a medium that naturally affords collaboration in cyberspace, end users started evolving into authors assuming more creative roles of generating content and sharing it with others.

Web 2.0 has brought collaboration to the next level. New collaboration patterns illustrated in the use of blogs, wikis, and participation in social networks, such as Facebook allow end users to develop their own content and contribute to the wealth of publicly available information. However, the complexity of the content has not gone further than text and multimedia. A big step for increasing the complexity of artifacts that end users collaboratively develop would be to add interactivity to the artifacts; that is, add behavior to the artifacts, to enable them to react to users' actions. This would elevate the creative process to end-user development (EUD) [3] and afford the authoring of more complex artifacts. We believe that interactive applications such as games and simulations are the next generation of content through which end users can collaborate. These applications enable end users to create content that is no longer passive and instead can be seen as active participants in cyberspace, moving from being information consumers to information producers [2].

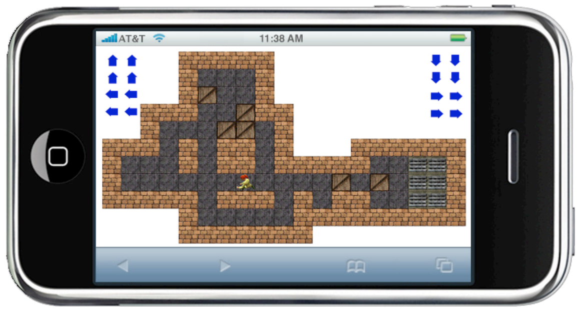

Figure 1. Ristretto ^{Mobile}: an iPhone running an end-user developed game

Handheld devices are the next collaborative platform. Unique features of handhelds such as portability, social interactivity, and connectivity [8] make them an ideal platform for users to collaborate through different content types such as text (instant messaging, blogs, and wikis), images (photo sharing sites such as Flickr), and movies (video sharing sites such as YouTube). Moreover, handheld devices have shown promise as a platform for technology-enhanced learning in educational environments [1].

However, there is very limited end-user support for authoring and running more complex end-user developed applications on handhelds. This lack of support is mainly due to specific properties of handhelds, leading to some challenges [7]:

• User-interface constraints of handheld devices, such as small screen size and limited input devices and modalities;

• Hardware limitations, such as slow processors, limited memory and storage, and restricted operating system functionality;

• Lack of a unified programming platform due to the diversity of hardware used to build handhelds.

We have created Ristretto ^{Mobile}, a Web-compliant framework for executing end-user developed applications on handheld devices. The applications are developed using AgentSheets [5], an end-user development environment, which enables end users to develop their own games and simulations using a graphical rule-based programming language. Ristretto ^{Mobile} has a Web-based architecture, which not only enables the seamless integration of end-user developed applications into Web 2.0 applications, e.g., social networking applications, but also addresses the lack of unified programming platform for handheld devices. We have succeeded in running Ristretto ^{Mobile} on iPhone, iPod Touch and Nokia N800. Figure 1 depicts an iPhone running an end-user developed game called Sokoban using Ristretto ^{Mobile} in the iPhone's Safari Web browser.

Our ultimate vision is to build an infrastructure that enables end users to collaborate through interactive applications, such as games and simulations. In order to reach our ultimate vision, we first need to provide the required infrastructure for large-scale collaboration by taking end-user development from desktops to the Web, using Web-compliant technologies such as JavaScript and AJAX. Ristretto ^{Mobile} is a feasibility prototype of *collaborative end-user development*, which at this point only provides the execution engine. Later on, the authoring tools and collaboration aspects will be added on top of the execution engine, to enable end users to build the applications and interact with them collaboratively.

In this paper we discuss previous work that led to Ristretto ^{Mobile} and the design goals for Ristretto ^{Mobile} for end-users, we present the architecture, explain the implementation, and discuss the challenges and future directions.

2. History

Early on in the development of AgentSheets, it became clear that users not only wanted to make simulations and games, but they also wanted to share them on the Web. For the most part, users did not have strong preferences regarding the underlying technology for publishing their projects, as long as the game experience on the Web matched the one using the authoring tool.

We employed the notion of behavior processors [4] as an analogy to word processors for a layered architecture between high level authoring and low-level implementation. In the case of word processors, few people would edit Postscript files directly as a means for creating a document. Instead, they use tools such as LaTeX or MS Word, allowing them to operate at a much higher level of abstraction. This also allows them to generate different output formats such as Postscript or Portable Document Format (PDF) without having to change their document. In the same spirit, we built a behavior processor called Ristretto [4] to automatically turn AgentSheets projects into low-level program representations.

The first version of Ristretto turned AgentSheets projects, including agent behaviors and media components such as images and sounds, into complete Java applets embedded into Web pages. The Java version of Ristretto includes a bytecode compiler that turns agent behaviors directly into very efficient Java class files. Other versions of Ristretto produced different output formats such as Flash movies. We also experimented with creating simplified Java runtime environments that could run on handheld devices such as HP iPAQs. However, to receive sufficient Java support at that time, we had to use SavaJe, a Java-based OS that had to completely replace Windows CE on iPAQs, something that users were not willing to do.

To this day, the original version of Ristretto is the best choice for demanding simulation and game applications. The Ristretto-generated output runs very efficiently on Java virtual machines because it no longer needs to interpret behavior. Flash, initially just an interpreted language, is gradually catching up however.

Ristretto ^{Mobile} is a natural progression of this work. At this stage it is a feasibility prototype exploring how well complex game and simulation content can be mapped onto an AJAX-based Web application framework. Specifically, it explores how well JavaScript is suited to implementing compile-time and run-time tools for end users. JavaScript is not well known for its performance. Only now, with the relatively recent introduction of essential JavaScript framework extensions such as the 2D Canvas, can we hope to be able to build Ristretto ^{Mobile} efficiently.

3. Ristretto ^{Mobile}

Ristretto ^{Mobile} is the execution engine for running end-user developed applications on handheld devices. The architecture for Ristretto ^{Mobile} has to be flexible enough to allow building the authoring tool and eventually adding the collaboration aspects on top of the execution engine. Moreover, Ristretto ^{Mobile} is especially designed for handheld devices, which needs specific considerations at the architectural level.

3.1. Design Goals

In developing Ristretto ^{Mobile}, we have considered the following design goals:

- ***End-user developed applications as collaboration content:*** our ultimate vision is to build an infrastructure that enables end users to collaborate through complex interactive applications. Therefore, Ristretto ^{Mobile} has to be consistent with the current collaboration paradigms on the Web, and should be as integrated as possible with Web 2.0 technologies such as JavaScript and AJAX. To reach this goal, Ristretto ^{Mobile} has to be a Web-based framework, allowing end-user developed applications to run in the Web browser.

- ***Customization for handheld interaction:*** given the interface and interaction modalities of handheld devices, such as small screen and limited input devices (e.g. only a touch screen with one physical button on iPhone), implementation should be especially customized to facilitate user interaction with such devices.

- ***Customization for handheld resources:*** limited hardware resources force the proposed architecture to be implemented as a lightweight application that requires low processing power and memory.

3.2. Architecture

To reach above goals, we conceptualized Ristretto ^{Mobile} as a framework for generating and running end-user developed applications in Web browsers of handheld devices. With Web-based execution of applications, not only do we achieve a high degree of cross-platform execution for all the digital devices that are capable of running a Web browser, but we also provide the preliminary requirements for the seamless integration of end-user developed applications with Web 2.0-based social and collaborative environments such as social networking tools.

To generate and run a Web-based end-user developed application, we need to transform the original application, i.e. the application developed by the end-user using AgentSheets, to its equivalent version that is executable by technologies supported by modern Web browsers such as JavaScript and Flash, or the external Web-browser plugins. The latter would require building a plugin for a specific browser that executes the end-user developed application inside the Web pages.

Ristretto ^{Mobile} transforms an end-user developed application to its equivalent JavaScript program, enabling it to run directly in all the JavaScript-enabled Web browsers. Therefore, without the need for a browser-specific plugin, it preserves a high degree of browser and platform independence, and also avoids any additional running overhead. This also keeps our application as lightweight as possible to meet the limited hardware resources of handheld devices, and also eases the integration of Ristretto ^{Mobile} with other Web 2.0 applications.

Figure 2 depicts the Ristretto ^{Mobile} architecture for generating and running end-user applications. The application retriever is responsible for fetching and synchronizing the application files from the Web server using AJAX calls, or simply reading application files from local file system. The application transformer is a compiler written in JavaScript that takes application files as input and generates the equivalent JavaScript code for running the application in the Web browser on the fly. The runtime engine consists of the required infrastructure including data structures and routines necessary for providing the runtime version of the end-user development environment rewritten in JavaScript.

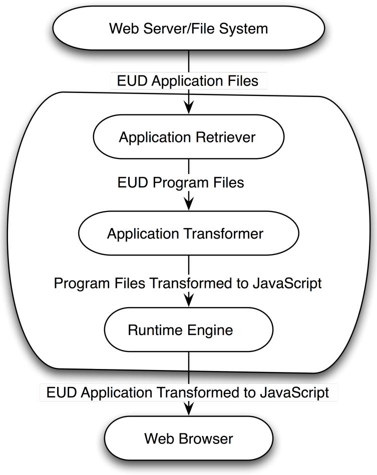

Figure 2. Ristretto ^{Mobile} execution engine architecture

In the proposed architecture for the execution engine, the entire application transformation process takes place on the client side. Client-side transformation allows us to build the authoring tool completely on the client side on top of the execution engine. Therefore, there would be no need to connect to the server for transformation when the user changes some parts of the application. If the transformation happened on the server, each client would need to connect to the server upon any change, Therefore, server-side transformation could become a performance bottleneck if the number of clients and changes increases. Client-side transformation distributes the transformation process load among the clients to ensure the scalability of Ristretto $^{\text{Mobile}}$. Another advantage of the client-side transformation is that users will be able to save the copies of end-user developed applications on their machine and not only execute them but also modify and rerun them, even when they are not connected to the Internet.

3.3. Implementation

We have implemented Ristretto $^{\text{Mobile}}$ to run applications developed using AgentSheets. AgentSheets is an end-user development environment for building interactive applications such as games and simulations. The world in which the applications run, called a worksheet, is essentially a grid containing stacks of agents. Agents are depicted as 2D images. Each agent is programmable by ends users using a rule-based language called Visual AgenTalk [4]. Agent behaviors contain collections of methods. Each method contains rules, which in turn are constructed from predefined conditions and actions.

Ristretto $^{\text{Mobile}}$ consists of the following components:

Application Retriever is responsible for downloading AgentSheets project files from the Web. The application retriever is basically a function that takes a URI as input and downloads the file content using XMLHTTPRequest. The application retriever is called by the application transformer whenever a new file has to be read and transformed. Later on, for adding collaboration features to the Ristretto $^{\text{Mobile}}$, the application retriever will contact the server once in a while to receive the changes in the end-user developed application files and pass it to the application transformer.

Application Transformer takes an AgentSheets end-user developed application URI from the user and transforms the application to the equivalent JavaScript program. First, it generates one JavaScript object per AgentSheets agent and transforms each method in the agent to the corresponding JavaScript method in the related object. Each condition/action used in the rule-based behavior translates to a call to the runtime engine. After translating all the agents, the application transformer reads the application's worksheet information, instantiates an object per each agent in the worksheet, and puts the object in the same place in the agent stack structure in the runtime engine.

Runtime Engine consists of all the static parts of the framework, such as initialization for running the application, required data structures, and a library of conditions/actions. Initialization involves setting global variables, creating data structures, such as a 2D matrix of stacks where the objects will be placed, implementing the agent super class, and methods for drawing the agents in a canvas in the Web browser, Runtime engine is also responsible for catching input events, e.g., key presses and mouse clicks, and sending them to the transformed end-user developed application. In addition, each of the predefined conditions/actions in AgentSheets is implemented as a separate method, which is called by the objects generated by the application transformer. All the static code is produced as a single JavaScript file.

3.4. Experiments and Challenges

The Apple iPhone and the Nokia N800 are two concrete examples of network-enabled handhelds with full Web browser support including JavaScript, AJAX and Canvas. The iPhone is a handheld device with innovative features including Multi-Touch and Safari Web browser. The iPhone has no physical buttons, except a "home" button. Users interact with the device via a multi-touch screen and a virtual keyboard covering about half of the screen that pops up so that user can tap keys. Moreover, it is only possible to use the keyboard in the browser if there is a textbox to type in. Therefore, there is no keyboard available while the user deals with the Canvas. The Nokia N800 is called an Internet tablet, running a version of Debian Linux operating system. We ran a Mozilla based browser on it to make sure it supports the Canvas. It has a physical keyboard and a touch-screen. But as with the iPhone, the keyboard does not work with the Canvas.

From the beginning, we intended to make Ristretto $^{\text{Mobile}}$ run on the Apple iPhone and Nokia N800. We used a number of interactive end-user developed games using AgentSheets, including Sokoban and Frogger, to test our application on these handhelds. Development challenges arose as follows:

Interaction challenges: AgentSheets applications mostly interact with users via the keyboard. For instance, in most games, the user controls the main character via the cursor keys. Due to the limited input modalities for the mentioned handhelds, e.g., absence of keyboard when dealing with Canvas, keys used in an application should explicitly manifest themselves on the screen so that the user can tap on them. This can be easily done in the AgentSheets authoring tool by creating button agents, which are triggered using click events instead of key events. Later on, instead of requiring the end-user developer to manually transform the key interactions to click triggers, automating keyboard simulation on the handhelds will be considered. Thus, users will not need to adapt the application so that the AgentSheets native applications will be able to run on the handhelds.

Implementation challenges: Considering the low CPU speed and limited memory of the handhelds, the transformed version of the application has to be lightweight. Therefore, we needed to take performance into account early in the design phase. Reducing the number of calls of time-consuming functions such as random and time, and avoiding redundant data structures are examples of design guidelines we followed to make the implementation work efficiently. Due to the slowness of the Canvas calls, e.g. the *DrawImage* method, we designed an optimized drawing algorithm to redraw only those parts of the scene that have changed since the last execution cycle.

4. Future work

The current version of Ristretto $^{\text{Mobile}}$ only includes the execution engine that retrieves and executes AgentSheets applications in the Web browser. Adding a Web-based authoring tool is the next step. The main difference of the Web-based authoring tool from its desktop counterpart will be the collaborative authoring and execution of the end-user developed applications. Though the execution engine is not directly involved in collaborative aspects of end-user development, it is flexible enough to build collaborative features on top of it using AJAX calls to synchronize with the server.

We will also investigate the integration of the collaborative Ristretto $^{\text{Mobile}}$ into other Web 2.0 applications, particularly social networking applications, e.g., Facebook.

Raising the ceiling of end-user generated content running on handhelds with 3D applications is another direction we are exploring. Android is a software stack for mobile devices that includes an operating system, middleware and key applications. It supports 2D as well as 3D graphics based on OpenGL ES 1.0. OpenGL ES is a cross-platform API for 2D and 3D graphics on embedded systems including consoles, phones, appliances, and vehicles. Moreover, the need for 3D graphics in the Web is getting increasing attention. Standards such as X3D have been developed, but there is no concrete support in Web browsers yet. Many browsers now support Canvas and probably in the near future a 3D context based on OpenGL ES API. These emergent technologies will enable 3D end-user development environments (e.g., AgentCubes [6]) to run on Web browsers of handheld devices.

5. Conclusion

Handheld devices are the next collaboration platform for end users, going beyond static content to fully interactive applications such as games and scientific simulations. Such end-user developed applications need to run efficiently on handheld devices. However, interaction and resource limitations of handhelds require specific support for running end-user developed applications. In this paper, we introduced Ristretto $^{\text{Mobile}}$, a Web based framework for generating and running complex interactive end-user developed applications on handheld devices efficiently.

6. Acknowledgements

This work is based in part upon work supported by the National Science Foundation under Grant Number IIP-0712571. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

7. References

[1] T. W. Chan, J. Roschelle, S. HSI, Kinshuk, M. Sharples, T. Brown, C. Patton, J. Cherniavsky, R. Pea, C. Norris, E. Soloway, N. Balacheff, M. Scardamalia, P. Dillenbourg, C. K. Looi, M. Milrad, U. Hoppe, M. Nussbaum, R. Mizoguchi, H. Ogata, R. McGreal, and H. v. d. Merwe, "One-to-one Technology-Enhanced Learning: An Opportunity for Global Research Collaboration.," *Research and Practice in Technology Enhanced Learning,* vol. 1, pp. 3-29, 2006.

[2] G. Fisher, "Beyond "Couch Potatoes": From Consumers to Designers". *In proceedings of the 3rd Asia Pacific Computer Human Interaction Conference,* pp. 2-9, 1998.

[3] H. Lieberman, F. Paterno, and Wulf V. (Eds), *End-User Development*, Kluwer/Springer, 2005.

[4] A. Repenning and A. Ioannidou, "Behavior Processors: Layers between End-Users and Java Virtual Machines", *in proceedings of IEEE Symposium of Visual Languages*, Capri, Italy 1997, pp. 406-413.

[5] A. Repenning and A. Ioannidou, "Agent-Based End-User Development," *Communications of the ACM,* vol. 47, pp. 43-46, 2004.

[6] A. Repenning and A. Ioannidou, "AgentCubes: Raising the Ceiling of End-User Development in Education through Incremental 3D." *In proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing*, Brighton, United Kingdom 2006, 27- 34.

[7] J. Roth, "Seven Challenges for Developers of Mobile Groupware," in *Mobile Ad Hoc Collaboration Workshop* Minneapolis, 2002.

[8] E. Soloway, C. Norris, P. Blumenfeld, B. Fishman, J. Krajcik, and R. Marx, "Handheld devices are ready at hand," *Communications of the ACM*, vol. 44, pp. 15-20, 2001.