END-USER PROGRAMING FOR THE WEB WITH FREEDOM

Asaf Adi, Maya Barnea, Nili Guy, Samuel Kallner, Yoav Rubin, Gal Shachor
IBM Haifa Research Lab
Mount Carmel, Haifa University Campus, Israel

ABSTRACT

Freedom is a model-driven, web-based, end-user development platform that is optimized for business users. Freedom simplifies development by quickly focusing on the business goal at hand, using the visual attribute of the task as the development abstraction using WYSIWYG editing. End-user development quality concerns are addressed in Freedom through division of labor and task-related focus. To reduce barriers to adoption by business users, improve the end-user experience, and simplify integration, Freedom leverages Web 2.0 technologies such as AJAX and REST

KEYWORDS

Web 2.0, web development, WYSIWYG, HCI, MDA

1. INTRODUCTION

Web 2.0 positions the end-user as an active entity. Many Web 2.0 related sites and phenomena such as Flickr and blog sites would never exist without end-users contributing content, opinion, and decisions. Nevertheless, we have yet to see large scale end-user application development over the web and, to a large degree, inadequate tools are to blame. In the business arena, end-users are discovering that IT departments focus on strategic initiatives with requirement queues that are sometimes years long. This effectively puts business users in charge of their own solutions and gives rise to do-it-yourself (DIY) situational applications [Situational Applications, Wikipedia]. The prototypical member of this study's target audience, the business user, is an educated professional (e.g., accountant, HR personnel) whose main goal is business-specific. Business users have modest computer literacy (and interest) that mostly include the web and MS Office, hence we can infer only basic computer experience that includes using a wizard to generate something new; interacting with spreadsheets, documents, and forms; and using drag and drop to rearrange items on the screen.

Freedom is a model-driven, web-based development platform that addresses the business users' development needs by supporting the concept of Goal Driven Development (GDD) and providing effective, web-based, development tools that assist business users developing DIY web applications. Freedom leverages the two complementary domains of web engineering and end-user programming.

Web engineering aims at improving the production, maintenance, usability, and other engineering aspects of web site development. Being a model- and template-driven web creation platform, Freedom can relate to previous studies in this area.

Model-based web development can be represented by efforts such as WebML [Ceri, 2000] and simple template generation tools [Turau, 2002][Zdun, 2002]. WebML offers the developer a full scale modeling language that can be used to model a web application end-to-end (content, page flow, database interaction, etc.). Once the model is defined, an application can be generated.

Template-based generation tools [Turau, 2002][Zdun, 2002] and, to some extent [Ruby On Rails] and other open source frameworks, present more lightweight approaches to using models and templates to construct applications, where the developer designs a set of templates that are employed when creating the content of the pages flowing back to the users. Some of the studies (e.g., [Turau 2002, Ruby On Rails])

describe a formal definition of a very lightweight model that validates and manages the data that the user communicates to the databases.

End-user programming aims at empowering the users and allowing them to produce applications. There is an obvious conflict [Harrison, 2004] between the web site's engineering quality and EUP since most users lack the desire (or self-discipline) to produce a well engineered application that has been thoroughly tested. Even so, the value associated with unleashing the user's productivity is high enough to justify the accompanying quality risk.

In the context of EUP and the web, we can see efforts in FAR [Burnett, 2001], CLICK [Rode, 2005], WebSheets [Wolber, 2002], and FlashLight [Rode and Rosson, 2003] to allow users to develop web applications without writing code. Some of these studies—e.g., CLICK [Rode, 2005]—are web-based, and so users do not need to install anything to start developing. FlashLight [Rode and Rosson, 2003], a Flash-based development environment, introduced the useful concept of programming at runtime.

Studies such as DENIM [Newman, 2003] and Whyline [Ko and Myers, 2004] are associated with the intersection between web engineering and EUP. DENIM [Newman, 2003] allows end-users to sketch the web site, its organization, and navigation and can be considered a beginner's design tool. Whyline [Ko and Myers, 2004] assists end-users in asking the right questions during the debug phase.

There are also studies on the needs and mental model of non-programmers (web masters) when developing applications [Rode et al, 2004]. It should be noted, however, that these web masters differ from business users in their computer literacy and attitude.

We first describe GDD, delve into Freedom—a tool that actualizes GDD—and present a sample Freedom-based EUP tool. We conclude by presenting future directions and summarizing our findings.

2. FREEDOM

Most of the EUP tools reviewed in the previous section facilitate end-user programming by eliminating the need to write code. The users are provided with controls such as pages, buttons, links, input fields, rules, and database records that they can use to compose their applications. However, for business users, this approach is too demanding.

The level of abstraction offered to the users by EUP tools is not sufficient. Most of today's EUP tools require business users to consider details such as navigation rules, database records, data validation, and formatting. These details are unrelated to the user's goal. Second, the level of functionality offered by the EUP controls is too low. Business users would rather specify an input field using the type of data that is about to be collected (e.g., a phone number) and not as simple text field input with an associated validation rule. This is especially apparent with complex input types such as address and person name that usually include several correlating input and output fields.

Freedom Provides support for Goal Driven Development (GDD), a technique that puts the user's goal at the center of the development experience. Freedom is designed to appeal to the average business user. it requires no education and no programming skills, cumbersome download or installation of any new software, enabling users to focus on the problem at hand. A typical business user has limited experience using software "wizards," documents, spreadsheets and forms, and knows how to create items on a "canvas" and later rearrange them using basic drag and drop functions. With this common prior experience, the user can proceed immediately to create applications for any business task using Freedom, including Human Resources, Finance, Sales or Marketing, that automate common manual processes, such as the collection and analysis of information. Consequently, the delays encountered when a typical employee issues an application development request through an IT department are eliminated. As more business activities move online, there are more occasions to ask the IT department for help, but if you have to wait a lengthy period of time for support, or you don't have an IT department, you can save time, money and a lot of aggravation by quickly creating your own online business applications for just about anything.

2.1 Goal Driven Development

We define GDD as a development process that sets the user's end goal at the center of all activities. The end goal drives the development experience and provides the abstractions and metaphors that are used through the development process; all "non goal" related details are hidden (especially technical details).

The importance of GDD can be explained through the "paradox of the active user" [Carroll and Rosson, 1987]. Users are motivated to start quickly and complete their immediate task; they don't care about the system as such and don't want to invest time up front in getting established, setting up, and wading through learning packages. Moreover, most business users do not care how a certain application is implemented as long as they achieve their goal with a minimum of fuss.

One conclusion that can be drawn is that within the context of business users, it is better to provide a tool that lets business users disregard details (such as application layout, page flow, validation rules, and databases) and focus instead on their goal; hence GDD. But then, how can we know if such a tool covers enough of the business user's problem space?

To answer this question we cite FlashLight [Rode and Rosson, 2003], where is was found that about one-third of planned web applications can be addressed by EUP tools with proper data storage and retrieval support and another 40% can be satisfied by a set of five tailorable [MacLean, 1990] applications. These findings, while not directly related to business users and their applications, lead us to believe that it is possible to identify a set of high level goals that cover enough of the problem space explored by business users in their development efforts.

Now that GDD is defined, and some evidence indicates that it is useful in assisting business users, we can look into how Freedom makes GDD a reality.

2.2 Freedom: Realizing Goal Driven Development

Clearly, since GDD requires development abstractions and metaphors to be extracted from the end goal, we need a set of goal-specific tools (and not just one tool). For this reason, Freedom does not focus on producing a single tool, but instead on providing a platform that allows for quick development and consumption of goal-specific GDD tools. To this end, Freedom provides the following elements:

- Mechanisms to manage and define application templates. Application templates are developed by programmers and provide all the means required to design and generate an application that covers a single goal. Each application template is composed of a UI editor component that is used by the user to develop the application and an application generation component that emits, compiles, and provisions the application.
- An AJAX-based client-side application development and management shell. The user uses the shell to develop new applications and load the UI component associated with a specific template whenever needed.
- A model definition. The model represents the minimal information that is required to develop the application and can be annotated and extended in a template-specific manner.
- A set of [REST] services that can be combined to form the application's backend.

2.3 Developing with Freedom

With Freedom, business users start the development session by opening the Freedom development shell in their browser. The user is taken through a series of wizard pages for specifying simple details such as an application name, the look and feel, and the application template to apply. Once the application template is selected, the Freedom shell loads the template's UI and presents it to the user. At this point, the user leverages the goal-specific UI to develop the application. Freedom's architecture is depicted in Figure 1.

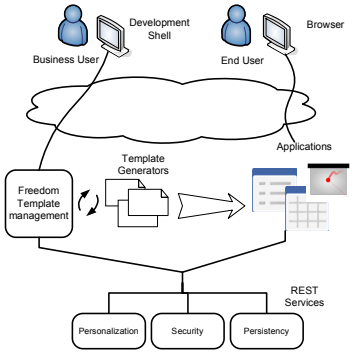

Figure 1: Freedom components and architecture

As the user develops the application, the Freedom development shell (with the template UI) implicitly builds a model that encapsulates the key attributes of the application. In Freedom, application models are concise, highly declarative, and tend to provide information such as the names and attributes of the business objects to be manipulated by the application. The amount of required information held in the model is relatively small. Details such as page structure and navigation, database schema, and template-specific business logic are embedded in the template and do not manifest themselves to the business user during the development phase.

Once the user finishes developing the application, he can save and publish it. Publishing an application triggers the execution of generators associated with the application template. The generators get the application's model as input and are then responsible for provisioning instance of the new application.

In their simplest form, the generators can implement the application UI elements by emitting HTML, JavaScript, and CSS resources that leverage the Freedom stock REST services. However, while most data entry applications can be implemented in this simple fashion, application templates may also go the route of leveraging a tailorable application and serving it to the user.

The end result of the development process described in this section is a realization of GDD. With the goal-specific editor, users can focus their attention on their end goal, disregard all details, and create new applications within minutes.

2.4 Freedom Forms – A GDD Tool Example

To better understand how the GDD UI can look, we explore an interesting class of applications and a possible GDD tool for creating form-driven, database-dependent applications.
Forms are central to today's businesses and business professionals spend a great deal of their time in completing and reviewing forms. The importance of forms led to the emergence of the [XForm] standard, yet, to date, business professionals are not creating their own form applications.

The form application template in Freedom bridges this gap by providing the business user with a friendly, WYSIWIG, form development environment, as presented in Figure 2.

Figure 2: Form application development UI

The UI visible to the user is focused on the goal, that is, the form. There is no place where the user can define Submit buttons, data models, or mappings in a database. The user simply designs the fields on the set of forms that constitute the application from the user's point of view. It is the responsibility of the form editor and the Freedom shell to deduce the required details automatically.

Recently we have explored the right GDD metaphors and abstractions for human based workflows. A workflow application aims to provide end users with an easier way to orchestrate or describe complex processing of data in a visual form [Workflows, Wikipedia]. The development of such applications varies between coding of the workflow to writing of an XML process definition file (as done in IBM LCD [Markus, 2007]) or drawing of the usual circles, boxes, and arrows which describe the workflow process.

In this case we have also adopted the WYIWYG model, i.e. the application developer is able to see and adjust the actual from view in each phase of the flow. The transition between flow states is described by determining the next phase for each button. Figure 3 present the views visible to the developer creating a two phase workflow, with a start phase (on the left hand side) and an approve phase (on the right hand side). Each phase is described by a tab with a new form instance and thus enables maximal visualization of the flow as will be presented to the end user. When focusing on a button, the developer is able to determine what the next phase will be, by creating a new phase or choosing a pre-existing phase. For each button, on each phase, in addition to determining the next phase, the developer can also choose actions to be performed when moving to the next phase (e.g. what message will be displayed, who to notify on phase change, etc.).

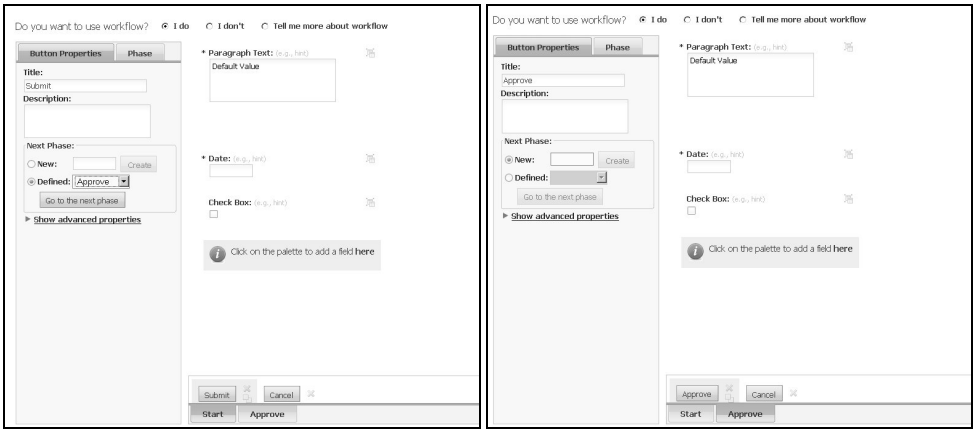

Figure 3. Workflow application development UI

3. CONCLUSION

In this paper we presented the concept of Goal Driven Development, an approach in which we address the development needs of business users and make the case for GDD, both from the user's perspective as well as its ability to cover enough of the business user problem space.
To support the vision of GDD and because GDD tools are goal-focused by definition, we presented Freedom, a platform that allows for quick development of GDD tools, and demonstrated a Freedom-based, situational application development GDD tool.

At this point, we are in the course of exploring GDD and its implementation to the depth, including more HCI aspects associated with GDD. At the architectural level, we would like to explore techniques for reducing the effort associated with producing new application templates. Finally, we are looking into techniques and tooling that can ease the creation of tailorable applications.

REFERENCES

Burnett M. M. et al, 2001. FAR: An End User Language to Support Cottage E-Services. In 1st IEEE Symp. on Human-Centric Computing Languages and Environments, Stresa, Italy, pp. 195-202.

Carroll, J. M. and M. B. Rosson 1987. Interfacing Thought: Cognitive Aspects of Human-Computer Interaction, MIT Press, Cambridge, MA.

Ceri S. et al, 2000. Web Modeling Language (WebML): A Modeling Language for Designing Web Sites. In Proceedings of the 9th international World Wide Web conference on Computer networks : the international journal of computer and telecommunications networking. Amsterdam, The Netherlands, pp. 137-157.

Harrison, W. 2004. The dangers of end-user programming. IEEE Software, Volume 21, Issue 4, pp. 5-7.

Ko, A. J. and Myers, B. A. 2004. Designing the Whyline: A Debugging Interface for Asking Questions about Program Failures. Proceedings of the SIGCHI conference on Human factors in computing systems. Vienna, Austria, pp. 151-158.

Markus Stolze et al. 2007, Developing Situational Workflow Applications with Lotus Component Designer, IBM

MacLean A. et al, 1990. User-Tailorable Systems: Pressing Issues with Buttons. In Proceedings of the SIGCHI conference on Human factors in computing systems. Seattle, Washington, United States, pp. 175-182.

Newman M. W. et al, 2003.. DENIM: An Informal Web Site Design Tool Inspired by Observations of Practice. In proceedings of the CHI conference. Fort Lauderdale, Florida, pp. 259-324.

REST: Representational State Transfer. http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm, accessed August 2008.

Rode J. et al, 2004. End Users' Mental Models of Concepts Critical to Web Application Development. Processing of Visual Languages and Human Centric Computing,. Rome, Italy, pp. 215-222

Rode J. et al, 2005. CLICK: Component-based Lightweight Internet-application Construction Kit. http://phpclick.sourceforge.net, accessed August 2008.

Rode, J. and M. B. Rosson 2003. Programming at runtime: Requirements and paradigms for nonprogrammer web application development. Proceedings of the 2003 IEEE Symposium on Human Centric Computing Languages and Environments. Auckland, New Zealand pp. 23--30.

Ruby on Rails. http://www.rubyonrails.org, accessed August 2008.

Situational Applications, Wikipedia, http://en.wikipedia.org/wiki/Situational_application, accessed August 2008.

Stiemerling, O. et al, 1997. How to Make Software Softer - Designing Tailorable Applications. Proceedings of the 2nd conference on Designing interactive systems. Amsterdam, The Netherlands, pp. 365-376.

Turau, V. 2002. A Framework for Automatic Generation of Web-based Data Entry Applications Based on XML. Proceedings of the 2002 ACM symposium on Applied computing. Madrid, Spain, pp. 1121--1126.

Wolber D. et al, 2002. Designing dynamic web pages and persistence in the WYSIWYG interface. Proceedings of the 7th international conference on Intelligent user interfaces. San Francisco, CA, pp. 228--229.

Workflow, Wikipedia, http://en.wikipedia.org/wiki/Workflow, accessed August 2008.

XForm. http://www.w3.org/TR/xforms/, accessed August 2008.

Zdun, U. 2002. Dynamically Generating Web Application Fragments from Page Templates. Proceedings of the 2002 ACM symposium on Applied computing, Madrid, Spain, pp. 1113--1120.