Streamlining User Experience Design and Development: Roles, Tasks and Workflow of Applying Rich Application Technologies

Xianjun Sam Zheng, Mo Wang, Gilberto Matos, and Shaopeng Zhang

Siemens Corporate Research, Princeton, New Jersey, USA
{sam.zheng,mo.wang.ext,gilberto.matos,
shaopeng.zhang.ext}@siemens.com

Abstract. The adoption of Rich Application Technologies (RATs), such as Windows Presentation Foundation (WPF) or Adobe Flex, not only significantly enriches the user interface (UI) technology, but also can boost the collaborations among user experience (UX) specialists, designers and developers. Many books and plenty of online resources have described and discussed the technical capability and details of various RATs. However, how to effectively incorporate RATs into the process of UX design and development has not been systematically addressed. In this paper, we report our experience of applying RATs to develop several complex enterprise software systems. A new role, integrator, is introduced to support the communications among UX specialists, designers, and developers. We discuss the responsibilities and task assignments for each role, and propose a new workflow to streamline the design and development. We also discuss the challenges and the lessons learned from applying different RATs.

Keywords: User experience, UI design, development, rich application technology, WPF/Silverlight, Flex.

1 Introduction

One major challenge of delivering successful user experience (UX) is "what you design is not what you will get". During implementation, developers often fall short of reaching the desired design quality defined by UX specialists or created by designers. The reasons can be attributed to the limitations of traditional user interface (UI) technologies, but also to the unidirectional workflow among UX specialists, designers and developers. The introduction and adoption of rich application technologies (RATs), such as Microsoft Windows Presentation Foundation (WPF)/Sliverlight, Adobe Flex, Java FX, not only significantly enrich the UI development technology, but also can boost the collaborations among UX specialists, designers and developers.

Many books and plenty of online resources have described and discussed the technical capability and details of these technologies. However, how to effectively incorporate RATs into the process of user experience (UX) design and development

J.A. Jacko (Ed.): Human-Computer Interaction, Part I, HCII 2011, LNCS 6761, pp. 142–151, 2011.

has not been systematically addressed. In this paper, we report our experience of applying RATs to develop several complex enterprise software systems. A new role, integrator, is introduced to support the communications among UX specialists, designers, and developers. We discuss the responsibilities and task assignments for each role, and propose a new workflow to streamline the design and development. We also discuss the challenges and the lessons learned from applying different RATs.

1.1 Waterfall Process of Software Development

Traditionally, software design and development follow a waterfall process (e.g, Cusumano & Selby, 1997; Royce, 1970). Figure 1 shows a simple example of this process. Based on the analysis of user needs and requirements, UX specialists create sketches or wireframes to define the system layout and also the interactions among different components. Designers will work on the visual design or "look & feel", adding visual style and drawing necessary icons. These "static designs" or mock-ups, often in the format of JPG, PNG, or BMP, will be delivered to developers, who will program business logic and behaviors to make the UIs functional. Custom or nonstandard UI controls, even though they can be an integral part of novel user experience, are often dropped out or changed during development because it is too difficult or requires too much effort to implement using complex technologies, such as graphics device interface (GDI+). Furthermore, because the whole workflow is unidirectional or one-way, it is rather time and budget consuming if the design needs to be changed at the end of development cycle.

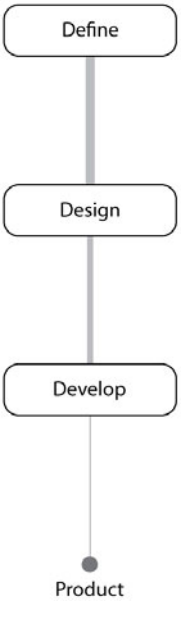

Fig. 1. An illustration of the traditional, one-way workflow for design and development

1.2 Model/View/Controller (MVC) Pattern

To support the concurrent design and development, the ideal process is to decouple presentation and user interaction from business logic and data access following the Model/View/Controller (MVC) pattern. Under MVC, the view manages the presentation or visual feedback to the user; the controller interprets the input or events from the user, such as mouse clicking or keyboard pressing, and manages the model information or updates the view, the model manages the data, responds to requests for information about its state from the view, and responds to instructions to change state from the controller (e.g., Krasner & Pope, 1988).

MVC pattern has been successfully applied in the web application development (e.g., Leff & Rayfield, 2001), where the view is the look of a HTML page, and the controller is the code that collects and generates dynamic data, and model is represented by the content, which is often stored in a database or XML files.

Several rich application technologies, such as WPF, Flex, Java FX, provide implementation support that leverages the MVC pattern. Two most representative examples are WPF and Flex, and both of them have mature and optimized tool support, which plays an important role in their successful adoption by the developer community.

2 Rich Application Technologies

2.1 Windows Presentation Foundation (WPF)

WPF, introduced in the Microsoft .NET Framework 3.0, features a new programming paradigm as an alternative to the traditional Windows Forms programming (e.g., Nathan, 2006; Petzold, 2006). WPF effectively permits the separation of UI from the underlying business logic using a new declarative language: Extensible Application Markup Language (XAML), which is based on XML. The creation of XAML intends to facilitate the communication and collaboration between designers and developers. For instance, detailed designs can be specified in XAML by the designer, and directly used by the developer in implementing the system. The development process can also start from the developer, who first creates components using default styles, then delivers these styles in XAML form to the designer, who then can style it and can completely redesign its appearance.

In WPF, styles define the appearance and interactive behaviors of the elements in an application. A resource is defined for re-use purpose. The external resource dictionary files allow designer and developers to re-use the same asset across multiple applications, providing a consistent user experience for different applications in a suite. Another unique capability of WPF is that similar code base can be used to produce stand-alone desktop application and browser application. This feature can dramatically reduce the efforts and costs for maintaining two different code bases for stand-alone or web-based deployments.

In addition, Microsoft also introduced the Expression Studio, including tools like Design and Blend. The idea is that designers will be using Expression Design, which supports vector based graphics. All the graphics and design generated by designers can be stored using the XAML format, and referenced from both the XAML GUI

specification, and from underlying implementation code. Several implementation languages, including C# and Visual Basic .NET typically provide the business logic implementation, whereas XAML delivers specification of the presentation layer. In the case of more advanced visual customizations, the programming languages can be used to explicitely manipulate the styles of visual elements, but even in these situations, the XAML based styles are often reused and dynamically associated with the control elements. Developers can use both Blend and Visual Studio to code XAML and C#.

Blend is an important tool of Expression Studio not only because it has comprehensive functionalities, but also because it has a full span of features in the product development. Starting from Blend 3, a new project type, SketchFlow, has been introduced into both WPF and Silverlight. SketchFlow enables UX specialists to visually define the UI architecture and navigation of the whole system. It also provides the low-fidelity theme for controls. The deliverable of SketchFlow is an application embedded in a navigation panel with notation and commenting tools for other people, such as stakeholders, to provide feedback. Blend can also generate a Word document template embedded with the design. This can also be used to facilitate the communications and discussions of the design details and specifications among team members.

2.2 Adobe Flex

Flex is a SDK (Software Development Kit) created by Adobe company for building highly interactive applications that can be deployed on Internet browsers installed with Flash Player plug-in or desktops installed with Adobe AIR. The UI layout and graphical appearance are described by MXML, a declarative XML-based language. The programming language used to define business logic is ActionScript which is an object-oriented language. Flex has included a rich library of basic UI components, such as buttons, list box etc., as well as some advanced UI component, such as data grid, data graph etc. Flex framework has its advantage in flexibility. For instance, Flex at its core is event-driven and has many mechanisms to incorporate MXML and ActionScript to implement complex behaviors. This flexibility makes it easy for developers to quickly build prototypes.

Flash Builder (formerly called Flex Builder) is an Eclipse-based IDE (integrated development environment) for rich internet applications (RIAs) and cross-platform desktop applications development with the Flex framework embedded in. Its main features include intelligent coding, interactive step-through debugging, and a WYSIWYG editor for the user interface layout and appearance design.

Another tool, Flash Catalyst, has been introduced in 2010, which aims to rapidly create expressive interfaces and interactive content. As an interaction design application, Flash Catalyst intends to be a bridge that can bring UX specialists, designers, and developers together. The tool is fully incorporated in the Adobe design suite and can import graphic assets created in Adobe Photoshop and Illustrator. It also can export a skeleton Flex/AIR project to Flash Builder, allowing developers to implement the business logic using Flex framework. However, the primary usage of Flash Catalyst is to create wireframes and interactive projects without writing code. For instance, it does not even support code editing, nor does it support two-way

workflow with Flash Builder. It does, however, support the two-way workflow with Illustrator and Photoshop. This allows designers continue working in Illustrator or Photoshop to refine design and their art work, which later can be imported into the wireframes in Flash Catalyst. We view Flash Catalyst as a tool primarily for UX specialists and partially for designers. Although it is a tool with limited functionalities and targeting at specific set of users, Flash Catalyst is easy to learn and used, especially suitable for people with less technical background.

3 Roles, Tasks and Workflow

In the past several years, we have applied both WPF and Flex to develop several complex enterprise software systems across various domains, such as building management, healthcare IT, and energy services. RATs were chosen as the major front-end development technology for a couple of reasons. First is due to their flexible deployment model to either desktop or web browser. Second, the separation between UI presentation and functionality is suitable for the collaboration among different teams at different locations and time zones. Lastly, many new UI features offered by both WPF and Flex, such as animation, customer control, and data visualization, empower the designer to create a new and improved user experience.

Because of the large scope of the projects, multiple teams with different expertise, including UX specialists, designers, developers, were involved in the development. A new role, integrator, was introduced to support the communications among designers and developers. We first discuss the tasks and responsibilities for each role, as summarized in Table 1. We also discuss the workflow for the design and development, which is illustrated in Figure 2.

3.1 Different Roles

UX Specialist. The primary responsibility for UX specialists is to define and ensure product or system user experience. The tasks include 1) understanding and analyzing end-user's requirements, such as their needs, capabilities, and behaviors in their real-world environments; 2) then translating the analysis results of users' tasks, use cases, and workflow into UI layout and interaction models. The deliverables are wireframes, sketches, and storyboards. The typical tools UX specialists used are PowerPoint, Visio, etc, which are independent to any RATs. However, both SketchFlow and Flash Catalyst can be useful tools for UX specialists to create wireframes, storyboards, and some quick prototypes. Based on our experience, Flash Catalyst seems to be easier to learn, but provides limited tool support for collecting user feedback.

Designer. The main responsibility for designers is to design "look & feel" based on the specifications defined by the UX specialist. Designers are used to create static screens, mockups, and icons using primarily Adobe design tools, such as Photoshop or Illustrator. The introduction of RATs can have significant impact on designer's tasks. For instance, with WPF, designers no longer work on mockups or screens but the actual XAML code (even though Expression Design allows designers to produce assets like they do with Illustrator or Photoshop and then deliver them in XAML, the functionality for design is quite limited). The deliverables are XAML assets,

including icons, brushes, animations, appearance/skin, shape, color, and resources. Designers would mainly use Expression Design and Blend tools, which allow them to actually execute the implemented functional systems, and then to directly manipulate the visual resources and styles. The main benefit is that designers can have full control of their design, removing the need for a roundtrip communication and verification of the changes made by the developers. However, because both Expression Design and Blend are new to designers, some training for the tools is necessary. In fact, based on our experience, designers, especially those who don't have programming background, will have a steep learning curve for the Blend tool.

With Adobe Flex, the change for designers is minimal because designers are still using the same design tools, such as Photoshop or Illustrator. The main difference is that their deliverables are files in FXG format (Flash XML Graphics), which are XML graphics files, can be imported directly into Flash Catalyst or later into Flash Builder.

Developer. The primary responsibility for developers is to develop functionality. Their tasks include, 1) understanding and realizing the UI implementation architecture that supports the required user interaction patterns; 2) leveraging the existing reusable assets and the development team expertise toward a full functional implementation; and 3) building connections to the data source. They create a functional UI according to the specifications defined by UX specialist and integrator, but without spending extra time on customizing the look and feel. They implement interactive functionality, bind the UI elements to the corresponding data source, and implement event handlers triggered by the user inputs and actions. The deliverables for developers include skinless UI controls and data binding. In WPF, they use primarily Visual Studio tool and the programming language is .NET programming languages such as C# and VB.NET. Whereas, in Adobe Flex, they use Flash Builder and the programming language is ActionScript.

Integrator. The integrator is a new role, who should have extensive training in programming and also a good sensibility to design. In the early design phases, the integrator needs to inform the UX specialists and designers of the specific constraints inherent in the selected implementation technology (e.g., WPF or Flex) that they should be aware of. This type of interaction continues throughout the life of the project, with the adoption of additional architectural decisions and implementation constraints being discovered. His/her responsibility is to assign tasks (functionality implementation or skin design) to developers or designers. Finally, the integrator is heavily involved in the production of a complete system, including the designers' declaration of visual artifacts, and the functionality and behavior implementation of the UI, coded by the developer. In WPF, the major tool used by the integrator include is Expression Blend. In Flex, the integrator will be mainly using Flash Builder and Flash Catalyst.

3.2 Workflow

RATs support concurrent design and development, and the integrator works on synchronizing the concurrent specialized teams. Consequently, a new workflow is proposed to streamline the UX design and development as illustrated in Figure 1.

Table 1. A summary of different user roles and tasks, tools for WPF (rows with white background) and Flex (rows with light gray background)

Roles	Tasks	RATs	Language	Tools
UX Specialist	Define UI concept, create wireframes & sketches	WPF		PowerPoint, Visio, SketchFlow
		Flex		PowerPoint, Visio, Flash Catalyst
Designer	Design visual "look & feel"	WPF	XAML, styles & resources	Expression Design, Blend
		Flex	MXML, graphic assets	Adobe Illustrator, Photoshop
Developer	Implement functionalities & interactive behaviors	WPF	C#, VB.Net, XAML components	Visual Studio
		Flex	MXML/ActionScript	Flash Builder
Integrator	Define UI architecture, coordinate and integrate the activities of UX specialist, designer, and developer	WPF	XAML/C#	Expression Blend
		Flex	MXML/ActionScript	Flash Builder

UX specialists, designers, and developers no longer work in a waterfall or one-way workflow. In fact, defining, designing and developing can happen concurrently, and there are a lot of two-ways communications between UX specialists and designers, UX specialists and developers, and also designers and developers. However, these communication and collaboration need to be coordinated by the integrator, who will sometimes work closely with UX specialists to understand and implement the UI architecture (e.g., screen layout), sometimes work closely with designers to provide technical support to generate XAML or MXML code, and often work closely with developers to make sure functionality implementation can be seamlessly integrated with the design.

Note that both UX specialists and integrators are responsible for the UI quality, with UX specialists focusing on the design while integrators focus on the realization and implementation. Both UX specialist and integrator are talking to designers and UI developers; but the UX specialist has more influences on designers than the integrator; conversely, the integrator has more influence on the UI developers. The integrator is also responsible for the communications between the backend and the frontend development. Sometime, there are overlaps between designers and developers, especially for the UI control interaction and animation. Designers can directly create some animations using XAML or MXML; but for special animations, it is much easier to implement in C# or ActionScript by the developer. The integrator needs to anticipate these issues, and make decisions accordingly.

This new workflow, on the one hand, encourages the communication and collaborations for different team members with different expertise, on the other hand, ensures that the efforts from different team members, such as UX specialists, designers, and developers, can be integrated together to deliver the desired design quality. As the result, it can achieve the outcome of "what you design is what you get".

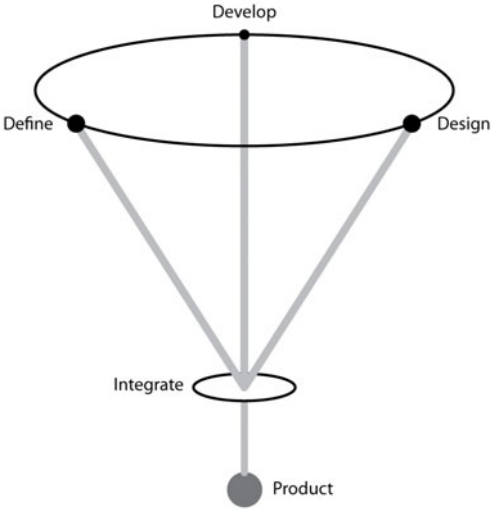

Fig. 2. An illustration of the new workflow for UX design & development

4 Lessons Learned and Discussions

Overall, RATs and their associated tools offer significant support for concurrent software development with multiple teams with different expertise. Our experience shows that with clearly defined roles and task assignment, as well as a new workflow, we can utilize the power of RATs to streamline the UX design and development. Nonetheless, there will still be significant challenges in developing advanced interactive system, even when fully leveraging the capabilities of RATs.

4.1 Non Technical Lessons

First, even though new technology is in place, it takes time for developers and designers to adopt the new process and development model. For instance, in WPF, instead of delivering static graphics or screen that they used to do, designer should deliver production artifacts, preferably in XAML format as a resource dictionary that can readily be used by the controls used in the implemented application. Developers should only focus on the functionality, and should pay no attention to the "look" of the UI, only to using the right controls that will use the proper visual skin from the resource definition. Secondly, the integration between designers and developers needs to be planned ahead rather than ad-hoc. Namely, the integrator should work

closely with UX specialist to lay down the UI architecture, and define the UI controls. Based on the analysis, the integrator shall decide whether a control template or data template is needed. Thirdly, the learning curve of WPF can be steep, especially for designers who have no programming training or experience.

4.2 Technical Lessons

Data binding. Both WPF and Flex generally create a good framework for separation of concerns between the UI designers and application developers, and allows the two groups to develop their artifacts independently, with or without tight integration of changes. This decoupling tends to break when the UI design tries to push the limit of the dynamic visual controls, where the controls need to be tightly coupled to the data bound objects for some aspects of graphical definition. In these situations, the data binding mechanism becomes a glue technology for the implementation of the appropriate object structures, and their mapping to the dynamically controlled interactive controls. For integrators, Blend provides more powerful tools for data binding to generate sample data of common types (e.g. name, address, email, phone number). In Flash Catalyst, the integrator would have to manually input the sample data.

Style (reuse control group). WPF and Flex provide a structured resource management framework in Expression Blend and Flash Builder, allowing the UI designers to create and replicate visual styles in resource dictionaries. While these capabilities enable the quick initial creation of visual resources, their effective maintenance becomes very time consuming if the stylistic dependencies are not managed by reusing shared aspects of the design elements. In general, the definition and maintenance of the shared and reused resource structures is the type of task that fits better with the software developers than with the UI designers' skillset, but it still requires a keen understanding of the visual dependencies as well. This aspect of style and resource management should be handled or overseen by the integrator who needs to have the knowledge of both the visual and structural aspects of the application architecture, as well as the planned or potential changes.

Non-standard behavior. Many components in WPF and Flex provide a standard behavior and allow wide customizations, both in their visual presentation and interactive behavior, as do a multitude of external component libraries. The customization capability often tempts UX specialists to try to define optimized interaction and behavior capabilities which provide a marginal benefit to the application user over the standard behavior. Unfortunately, changes in one aspect of behavior for complex user controls may require extensive rework and impact many other aspects due to the implementation dependencies. The actual cost and effort for such customizations in many cases may make them non-cost effective, particularly if the expected benefit is small. In addition, once a control is extensively customized and modified, it often needs to be maintained to preserve the customization with new releases of the base components. In general, only if a major competitive advantage of a system depends on modifying the standard control containers, can such a decision be justified.

5 Conclusion

The emerging technologies, RATs, are making it easier for UX specialists, designers, and developers to work collaboratively on developing new applications, by concentrating on their respective specialties and having a well defined integration interface. This interface is standardized on the technology level and does not map to the domain specifics of any application, so there is still much left to interpretation of the assets that are coupled over the interface, and to the common interface understanding between designers and UI developers. The management of this communication interface, which is both complex and project critical becomes the primary responsibility of the integrator that collaborates with the designers and developers in making sure the shared resources are defined and used in accordance with a common interpretation of the design and behavior requirements.

Acknowledgments. We thank Laurent Bugnion for valuable discussions and inputs. We thank Miao Wang & Nicola Vittori for helping create the visual illustrations.

References

1. Cusumano, M., Selby, R.: How Microsoft builds software. Communications of the ACM 40(6), 53–62 (1997)
2. Leff, A., Rayfield, J.: Web-App. Dev Using the M/V/C Design Pattern. In: Fifth IEEE Intl. Enterprise Distributed Object Comp. Conference (2001)
3. Krasner, G., Pope, S.: A cookbook for using the model-view controller user interface paradigm in Smalltalk-80. Journal of Object-Oriented Programming 1, 26–49 (1988)
4. Nathan, A.: Windows Presentation Foundation Unleashed (WPF). Sams Press, USA (2006)
5. Petzold, C.: Applications= Code+ Markup: A Guide to the Microsoft Windows Presentation. Microsoft Press, Redmond (2006)
6. Royce, W.: Managing the Development of Large Software Systems. In: Proceedings of IEEE WESCON, vol. 26, pp. 1–9 (August 1970)