Creating Effective RIA Interfaces

Grant Skinner, gskinner.com
Digital Media Innovation

Note: These notes were for a session at FlashForward New York 2003. To view the speaker slides for the session containing information more specific to Flash, please visit http://gskinner.com/talks/ffny2003/

Introduction //

The way users interact with the World Wide Web is constantly evolving. The current direction is towards more usable and functional sites which blur the line between desktop applications and web sites. This step forward is embodied by the term Rich Internet Application, which describes an online application or utility that includes a level of functionality and interface complexity formerly ascribed only to desktop applications. RIAs can be developed with a number of technologies, including Java, DHTML, and perhaps most successfully, Macromedia Flash. But regardless of the technology used to develop and deploy them, RIAs all share the common need for a usable, effective interface to present to the end-user.

As interface designers, we often fool ourselves into thinking we are "bridging the gap" between the user and her goals. In truth, regardless of how well an interface is designed, it will always duplicitously serve as the sole means, and largest barrier to a user reaching her goals. It is a UI designer's task to wrap functionality in a pretty package, incorporating a modicum of rainbow dividers and spinning logos from the fools* in marketing, while minimizing the hurdles a user must jump to successfully complete their tasks.

* Disclaimer: a few marketing people are not fools – my present, past, and future clients, for instance

Fig 1.1 – UI designer hubris (less the mackerel)

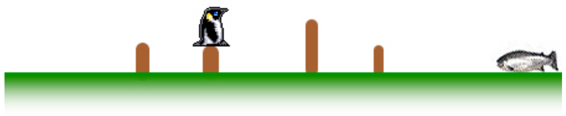


Fig 1.2 – UI designer reality (again, sans fish)

The three C's of good interface design //

All good things come in threes - this is an indisputable fact. It is also a fact that they are even better if they rhyme – failing that, alliteration is pretty cool too. As I'm not much of a rhyme smith, I have come up with "the three C's of good interface design", as invented by Grant Skinner.

These three concepts should be remembered, and applied to everything you do when creating interfaces. If you forget everything else in this session (I'd forgive you), don't forget this mantra: Context, Consistency, Communication!

Context

Determining context should form a huge part of the planning that goes into designing an effective interface. Before you sketch your first concept on a cocktail napkin, you have to understand the context of the project requirements, the operating environment, and the user base.

Once you begin the design, it is imperative to ensure that the options and content presented to the user are relevant to their context. Showing a user too many options at a time will confuse him, hiding too many will obstruct his ability to get anything done.

Consistency

Maintaining consistent interface standards throughout your application will reduce your users' learning curve for your application. They are able to learn approaches once, and apply them throughout the app. This applies to virtually everything you create in your interface development process: interface widgets, keyboard shortcuts, metaphors, icons, placement of buttons, error message text, help systems, etc. It is also usually a good idea to have a designated area of the interface that remains persistent, and largely unchanged throughout the app. This consistently available element serves as an anchor, and helps remind users where they are, what they're doing, and what their "big picture" options are.

In a similar vein, maintaining continuity (another "C" word) with "legacy" interfaces, such as Windows and MacOS not only greatly decreases the time it takes a user to understand an interface, it also helps foster trust through familiarity (it also means less work redesigning the proverbial wheel)

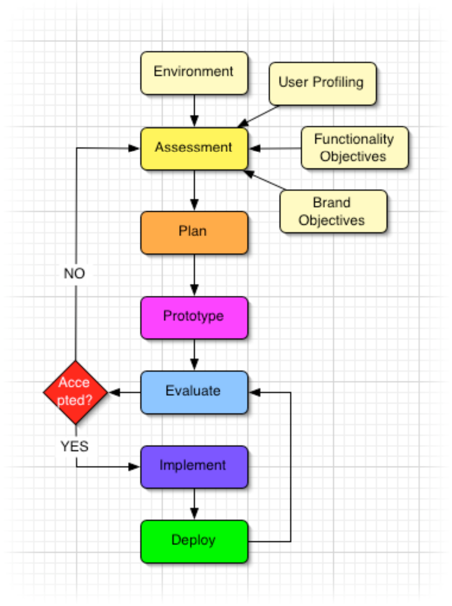
Communication

Ultimately, the sole task of any interface is communication. It must communicate to the user the available methods of interacting with the data, in a responsive and intuitive manner. It must then facilitate the user's requests in as transparent a manner as possible If it can't do this successfully, the user may as well punch SQL statements into the Oracle database on the back-end on their own. (Remember, we are trying to build a smaller wall, not a bigger bridge).

With RIAs we have two other areas of communication to consider specifically environmental and branding. Your interface's environment will most likely include a server and a browser, and you will often be called upon to help determine the most appropriate methods to communicate with each of these, to maximize the responsiveness, and usefulness of the application. Most online applications also have an important branding or marketing component (ie. Selling or advertising something, probably tiny cameras), and it will be up to the interface team how to incorporate these elements without sacrificing the usability of the interface

"The Process" //

As with all creative endeavours, most people approach interface design in different ways that they have found work best for them. Generally though, the most successful approaches have a number of commonalities, and as such, I will simply share my own process (which I have found to work very well), and leave it to you to incorporate or discard as much as suits your team. Consider it more a series of suggestions, than a blueprint.

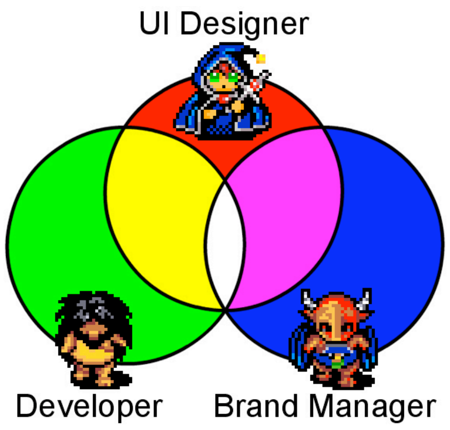


Virtually all projects are built by a team, even if it's just a single designer/developer, and his client. Because every member of this team should have input from project inception, we will start with…

Assembling a band of merry men (or women, or both)

As the UI designer, you are the centre of universe – seriously (don't get a fat head though)! Because your interface ties the aesthetic requirements of the brand to the functionality programmed by the developer, you can't possibly plan it without the input of the full team.

A typical application interface has two major components that affect its usability – layout and functionality. This combination necessitates the involvement of two professional groups in the interface creation process - designers and developers, respectively. Due to the commercial nature of most online content, RIAs frequently involve a third component – branding. This necessitates the involvement of a group or individual familiar with the deploying company's marketing strategies.



Too often, companies relegate the task of developing an online application to a team representing only one or two of the afore mentioned competencies. Creating a successful RIA demands the input of all three.

UI Designer

The UI designer is the star of the show (at least in this session). This may be an individual, or a team, with a skill set including visual/aesthetic design, UI design, and a reasonable understanding of both branding and the necessities of programming. Larger teams typically have a design lead, who is responsible for aesthetics and branding issues, and a UI specialist, who is concerned mainly with usability issues and acting as the liaison for the programming team.

Typical UI Designer

Developer

In a small shop, this is likely the same person as the UI Designer, which has the advantage of reducing the amount of communication necessary between teams (unless you suffer from a multiple personality disorder), but the disadvantage of requiring you to be very good at a lot of different things.

Developer (w/o coffee)

The developer is responsible for the technical side of the application. He will be doing the majority of the programming, and will be responsible for telling the designer what the functional requirements, and technical limitations of the application are.

Brand Manager

In many projects, the brand manager is simply the client. The brand manager dictates the branding goals and requirements of the application. This can include details like logo placement, font choices and color usage, as well as more broad reaching requirements like application functionality and purpose.

Brand Manager

The brand manager must understand (or know who to ask for) the full picture on what the RIA should accomplish, and how it will fit into the marketing efforts of the company deploying it.

In a perfect world, brand managers and developers never talk to each other, as they speak very different languages, and tend to have very different opinions on the end goal for any project. Unfortunately, this is not a perfect world, so ensure you have access to coffee and aspirin throughout the design process.

Now we have assembled our team of superheroes, and are ready to conquer the earth, or build an application, whichever comes first. But what are we building? That's a good question to answer before getting started, and so we move into…

Figuring out what the heck you're doing (aka assessment)

In this phase, you research and document the requirements, and the context (sound familiar?) of the RIA. You must determine the operating environment, the brand requirements, the functionality objectives, and develop user profiles.

Operating environment

Determine all of the environmental factors that will affect the interface design. Start by determining the method(s) of deployment. Will the RIA be deployed via the web, CDROM and/or a installable application? Once you've determined this, you can look at the factors specific to that environment. The obvious factors are visual - screen size, browser type and whether the interface will be displayed in a popup or a regular browser window. Less obvious environmental factors which must be considered are the minimum target processor speed of end users (to ensure the interface is responsive), the target plug-in version (4,5 or 6), the internet connection speed of the user, where the users' data will be stored (ex. Local shared objects or an online database). and the server environment (for instance, the availability of FlashComm might have a big impact on your design).

Branding requirements and objectives

Acquire, or develop a style guide for the company. This should include directions on issues like font use, logo placement, corporate colors and legal details (like where to put ™ and ® symbols). The brand manager (or client) should prepare a Project Objectives document, outlining what the specific goals and requirements of the application are. This document may already have been developed in the form of an RFP, or as the appendix of an overly long and binding contract you signed before starting the project (say goodbye to your firstborn).

Functional requirements

The programming team must take the requirements from the client/brand manager, and break it down into a listing of functional requirements. The result should be a document outlining all of the functional components of the application. Not surprisingly, the parts we are most concerned with are those that have an interface component. Unfortunately, this often isn't easy to determine with a brief glance, therefore, it is often a good idea to join the programming team in their cave (adorned with tribal Star Trek paintings, and arcane maps of the Java class hierarchy), and go through the listing together looking for interface requirements. Remember that they may be well hidden – for instance, a reference like "loads XML SOAP data from secondary server" will likely need some kind of status display.

User Profiling

In this stage, you will work with the brand manager to create a number of overviews of potential users. Identify each of the primary target demographics, and then write a short profile for each one, including any information that you consider relevant from a marketing or interface perspective. For example, you will generally want to include demographic information like age, gender, and earnings; technical information like OS, computer specifications, and browser of choice; and personal information like technical experience and comfort, and attention span. This will likely overlap to some extent with your Operating Environment documentation – feel free to revise as you go.

Now that you have a decent idea (in theory) as to what your goals are, and who is going to be using the application, you are prepared to begin…

Planning your interface

This is probably the most boring part of the interface design process (no joke). In this phase you go through your list of requirements (both functional and branding), and briefly write (yes, write, not draw) how you plan to satisfy those requirements and objectives. This lets you determine the best approach, and establish a consistent ("C") approach to your interface, before you start illustrating and become married to your own brilliant ideas. It serves as a reference when things go horribly wrong in the testing phase, and you have to explain why you chose to do things the way you did.

In parallel, you will want to write (or add to) a standards guide for the project. This guide should include everything from when to use a specific interface control, through icon usage to specific wording (is it a shopping basket, or a cart?) and error handling. This will be your bible as you design the interface!!

If this is your second or third time through this phase, it will consist mainly of reaffirming choices that worked, and revising choices that did not (you might want to document why it didn't work for future reference).

Prototyping

This is where most designers feel most comfortable. This is when you finally get to whip out the sketchpad, or fire up PhotoShop and start designing. You will likely run through this phase two to four times, preceding it with assessment and planning, and following with testing each time. Following are a few of the iterations you might go through before reaching a final prototype:

Sketches

Simple sketches that show the basic relationships and layouts of the interface elements make a good first step as they clearly show design intent, and are easy to modify.

Wireframes

Simple black and white comps done in Photoshop, Flash or Illustrator. These show the size and positioning of elements on screen, without overt concern for aesthetic details.

Paper prototyping

Paper prototyping is a crucial part of UI design. Paper prototypes allow you test interface layout AND functionality with everyone involved (brand managers, programmers, and sample users). A properly executed round (or rounds) of paper prototyping will save a huge amount of time and effort in the following design phases!

Comps

Static comps show the interface design in its full glory. At this stage, you can add color and aesthetic design to the wireframes, while ensuring you conform to the standards that were established in the Assessment and Planning phases.

Functional Mockups
In this optional phase, you add simple interactivity to your static comps using Flash. Functional comps demonstrate a similar level of functionality to paper prototypes, and can be guided, or interactive.

Once you have built a prototype, it's time to determine its suitability. On to…

Evaluating your interface

You have a prototype. You think it's great. It probably isn't… sorry, but its true, first round interface designs usually have a lot of room for improvement. And to prove that it isn't, there's nothing like a little testing.

Evaluation can take many forms, from the simplistic – simply showing and explaining the prototype to the major stakeholders - to the more costly and time consuming – such as full blown quantitative usability testing (usability metrics). Regardless of the type of testing, it is important to document the results properly, with reference to your original planning documents. Don't just focus on WHAT went wrong, try to determine WHY it went wrong by asking appropriate questions of the testers.

A common assumption made by interface designers (and techies in general) is that users are all idiots. While it makes for a great scapegoat for our own ineptitude, it is patently untrue – it's not that most users CAN'T figure out how to use complex interfaces, it's that they are NOT WILLING to do so just to see if your online store sells Nikes for 60 cents less than the next guy's. Keep this in mind while designing and testing, and you will go far in life (no guarantee implied).

If your application does not meet the objectives you originally documented, proceed directly to Assessment (do not collect $200), to determine if your criteria for success are reasonable, and then proceed through the process again. If it did meet the stated objectives then congratulations! You are ready to begin the arduous task of implementing and deploying the application, evaluating real world use, and starting all over for version 1.1. Hooray!