The Development Designer Perspective

Brent White and Lynn Rampoldi-Hnilo

Oracle Corporation, User Experience, 4500 Oracle Lane, Pleasanton CA 94588

Abstract. When creating a decision support tool to assist the selection of a software interface design it is crucial for the user experience team to understand how members of development make design decisions. How do developers think about design and what terminologies do they apply to the design process? Fifteen in-depth interviews were conducted with developers and product managers responsible for interface design decisions. Results indicated that members of development focused mainly on making new features consistent with their current applications and not innovating new interfaces. Development's design decision making process relied heavily on input from peers and senior staff. Development's terminology for categories that are factors in design (e.g., user type, user task, layout structure and data structure) differed from the version conceived by user experience. The synthesis of the two terminologies has created a basis for building a development decision support tool.

1. Introduction

Developer-designed interfaces for business applications are a reality of enterprise software. Applications often require thousands of complex screens, rich with functionality, that drive specialized work flows. Due to limited resources and the amount of time required to understand complex business domains, designers and usability engineers are only assigned to the highest priority applications.

Enterprise software refers to a suite of business applications that support common and strategic business activities such as employment payroll, managing customer relationships and planning future demand. The enterprise is generally separated into areas of related business activities and grouped into divisions. Each division has dedicated development teams that create designs and functionality independently from development teams that belong to other divisions. As a result, end users are often perplexed why the behavior of the interface varies for similar functionality across multiple applications. For instance, a supply chain management application uses a wizard to add a new customer while a financials application uses page forms and tabs to enroll a new client.

In a quest for consistency and improved design quality, an effort has been launched at Oracle to create a design pattern library. Design patterns are reusable and proven design solutions to common design problems. The initial effort has resulted in approximately 100 design patterns. The target audience for these patterns is design decision makers within the development organization. This group mainly consists of developers and product managers.

This paper addresses research that seeks to understand the development perspective on interface design in order to create an effective decision support tool. The decision support tool will assist developers in choosing the most appropriate design pattern.

1.1 Design Patterns

Christopher Alexander applied design patterns to architecture in the 1970s. Alexander *et al.* [1] defined patterns to be used in creating optimal living spaces from a house, neighborhood to community. An example pattern, entrance transition, described how to design the space between street and home. In this case, the pattern provided solutions such as changing the lighting, direction of the path, surface height and material in order to create transition from public to private. Alexander *et al.* [1] wrote:

> Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.

Design patterns became popular within the object-oriented software community in the early 1990s. Gamma, Helm, Johnson and Vlissides [2] applied design patterns for reusable object-oriented software design. More recently, design patterns have been adapted by the interface design community. Jenifer Tidwell [3] identified and described design patterns for Web applications and Yahoo! published their design pattern library online in 2006 [4].

Design patterns benefit development, user experience and end users. Development will become more efficient by having access to a library of proven design solutions. Designers will not have to constantly reinvent the wheel every time they work on a new feature. Instead of starting from scratch, a designer can apply patterns to straight-forward design problems and spend the majority of her effort on more difficult challenges. And in the end, increased design quality and consistency benefit the efficiency and satisfaction of end users.

Pattern Library Dilemma

Creating the pattern library was the first step. However, for this work to be effective, developers must be able to select the most appropriate pattern for their specific design problem. With so many patterns to choose from, how will development know how to choose the correct design pattern?

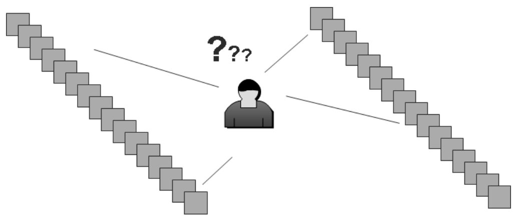

Fig. 1. Choosing from over 100 design patterns will be difficult for the development designer

The user experience group at Oracle decided to build a decision support tool to address this problem.

1.2 Solution: Decision Support Tool

Decision support tools have been applied effectively in domains such as aerospace cockpit design and business planning software where many factors change dynamically and a user's input must be taken into consideration. A decision support tool, defined by Turban [5], provides the type of interaction that will best support accurate selection of a design pattern: "…an interactive, flexible, and adaptable computer-based information system, especially developed for supporting … improved decision making. It utilizes data, provides an easy-to-use interface, and allows for the decision maker's own insights."

In order to inform our decision support tool, an effort was initiated to systematically describe the patterns using a faceted classification. Unlike hierarchical classifications of information that rely on knowing the top-level category, faceted classifications take a bottom-up approach and combine attributes of information, or facets, to describe the subject. A common usage of a faceted classification is for choosing a restaurant online. Users can search by price, location, ethnicity of cuisine or review stars. This system of information retrieval is flexible since users can get results by combining facets or searching only by one. It does not matter in what order the facets are combined.

The first pass at breaking down Oracle's design patterns into categories or *facets* resulted in the following:

- User Type
- User Task
- Data Structure
- Layout Structure
- Learnability

We included user type and task because they are key components for any design activity. "Understanding, representing and reasoning about the user's tasks provides an excellent means of accessing and organizing the knowledge the display designer needs [6]." The possible influence of back-end development on the front end was addressed by the data structure facet. The layout structure was related to design scope. Did the developer need to design a multi-page process, a single page or a section of a page? Learnability ranged from easy to difficult and was based on the degree of interaction complexities built into each design pattern.

Flexibility and adaptability are key characteristic of a decision support tool and vital for user acceptance. The decision support tool will fail if it is too rigid and forces developers to enter parameters that do not relate to their specific design problem. Users must be able to arrive at the correct design pattern from several paths or ways of thinking about the design problem. The requirements of an effective decision support tool provoked several questions. How do developers currently think about design? Does our terminology make sense? What are the key factors that affect design decisions? User experience did not want to create a decision support tool that was solely based on internal perceptions. At this point, we needed to conduct research with developers and product managers who we termed *development designers*; the key users of this tool. The main objectives were:

1. Learn about the development designer's process for building an application interface.

2. Understand how decisions that impact the interface are made.

3. Gain insight into user experience resource usage.

4. Identify differences in user interface terminology between user experience and development.

2. Methods

Two researchers conducted fifteen subject area expert interviews from March 3rd to 15th 2006. The interviews consisted of ten open-ended questions. Twelve interviews were face-to-face and three by telephone. Each interview lasted from forty to sixty minutes. A convenience sample was used. The sampling strategy aimed to include different roles, application divisions and organizational structures. Participants experienced in making design decisions were selected from the seven major divisions within the development organization. Five were developers and ten were product managers. Researchers selected participants from Oracle's headquarters and from two additional corporations that Oracle had recently acquired. This allowed for a greater spectrum of information to be gathered on the implementation of design processes due to different organizational approaches and corporate cultures. Two researchers alternated taking notes and leading the interview, switching after each participant. Both researchers established an informal and open atmosphere by not recording the interview, conducting each interview in private and ensuring participant confidentiality.

The interview first established a hypothetical development scenario that grounded the participant in a realistic context. The key topics of the interview were development process, the decision making strategy used when selecting a design, and terminology for areas that affected usage and design. The open-ended questions purposefully avoided words that user experience had employed during the faceted classification of the design patterns. Each use of a term that related to a facet was noted by the researchers. The learnability facet was not addressed by this research.

After completion of the interviews, the researchers discussed response themes for each objective, identified the most significant differences, and tallied every mention of a term that related to a facet. Terms that did not have at least two references were discarded. Similar terms such as *monitor* and *track* were combined into a single term.

3. Results

3.1 Objective 1: Learn about the development designer's process for building an application interface.

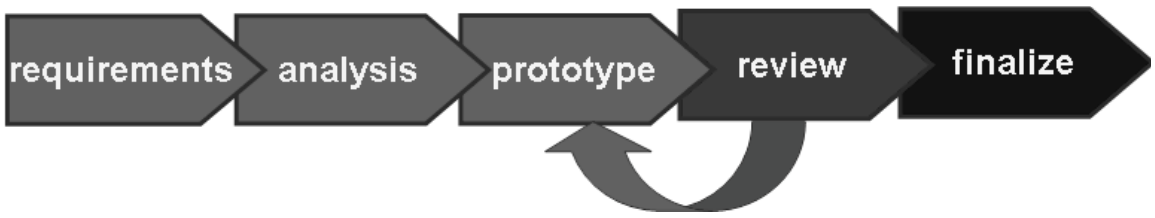

Fig. 2. High-level development process

After talking with our development colleagues, it became clear that developers followed a design process (Fig. 2). There was a separation of front-end and back-end programming. The front-end design took place first and drove nearly all back-end programming requirements. The user interface design was part of a long prototype cycle that often lacked a defined process. As a development product manager summarized: "There are lots of opinions, lots of reviews, lots of contradictions, and lots of should bes." Locking down the prototype was a significant milestone.

In general, developers were not looking for innovative ways to solve design problems. The overriding concern was to make it consistent with the existing product and to look for designs that have worked in the past. All participants spoke to some degree about this same process, which was not surprising, since the development organization does most of the design work at Oracle.

3.2 Objective 2: Understand how decisions that impact the interface are made.

Design decisions within the development organization were generally made in the following order:

1. Made it consistent with existing UI

2. Drew upon personal experience with Oracle applications
3. Asked peers or senior members of team for advice
4. Consulted Oracle's Browser Look And Feel (BLAF) guidelines
5. Investigated other applications outside their division or third party applications

Making the design consistent with the current application was the greatest factor when choosing a design for a new feature. Next on the list was the developer's personal experience with Oracle applications. Developers often repeated design solutions that had been applied to similar problems encountered in the past. Several development designers mentioned the term *intuition* when describing how they decided what design to choose. Experience and familiarity with Oracle applications were mentioned as factors. Many participants copied designs from mature products since these interfaces were tried and true. A senior product manager stated: "The more mature the product, the more reliable it will be."

When seeking in-person help, development designers usually approached peers or senior members of their development team. Junior developers requested design assistance more frequently than senior members. Each development team typically had a seasoned developer or product manager who provided a respected opinion concerning design. There was a strong clan mentality among development teams. Developers rarely went outside of the group to discuss design. The main contact with the user experience group was via its BLAF guidelines Web site. When peers or the BLAF guidelines did not lead to a satisfactory design, product managers sometimes consulted other members of the Oracle development organization or third party applications for new ideas and approaches.

3.3 Objective 3: Gain insight into user experience resource usage.

Development designers had mixed results when working with user experience resources or team members. Some of the positives were that user experience involvement had led to noticeable product improvement and that development felt more educated about user interface issues as a result. However, some of the drawbacks of user experience involvement were a perceived slowdown of the development process. Several participants said that user experience members did not have a deep enough understanding of the software functionality or product domain. Almost all were familiar with Oracle's internal online Browser Look And Feel (BLAF) guidelines. These guidelines are published internally by the user experience group. Usage of the guidelines was also mixed. Some developers consulted them often while others did not since they felt that the guidelines were difficult to search and often out of date.

3.4 Objective 4: Identify differences in user interface terminology between user experience and development.

The interviews provided considerable insight into how development approached design. In order to fulfill user experience's goal of building a decision support tool for development, we needed to understand the terminology, process and decision making strategies currently in place. During the interviews, we actively listened for terms that development used to classify users, their goals and layout structure. In addition, we were interested in knowing if data structure affected the interface.

User Type: Who are the different users of your application?

Most members of the development organization did not interact with end users. There were differences between divisions in understanding the profile of the actual end user. For instance, in one division, there was a general understanding of a project manager end user while in another division there was a much more nuanced understanding of the different types of end users (sales personnel vs. client services representatives) and their respective behaviors, those who preferred the keyboard (client services) vs. mousing (sales personnel).

In order for the decision support tool to be effective, we had to understand development's perception of the user. User experience classifications prior to this research led to approximately 10 user types that included analyst, executive and manager. Since these user types were specific to only a few divisions or had different responsibilities for the same user type across divisions, user experience reduced all user types to casual and power users.

Table 1. User experience vs. developer perspective. The left column displays the user experience terminology prior to the research. The right column presents the developer terminology as a result of the research

User experience perspective of user type	Developer perspective of user type
Power	Experienced
Casual	Casual/Self Service
N/A	Setup/Admin

All 15 of the interviews yielded information on the end user. The end users were typically characterized by experience with the application. Experience was gained by spending more time in the application. There was only one reference to *power user* and one to *expert user*. Classifying end users by an experience continuum rather than skill level differed from user experience's concept of the power user role. In retrospect, classifying end users by experience was probably a more realistic assessment since most enterprise users are paid to use software and will usually learn enough to get their job done but little more. The power user classification may be a case of projecting personal behavior on another set of users: most user experience members are power users of several software applications.

Development frequently mentioned the technical setup administrator as a user type. Enterprise applications require extensive setup since each institution or company that implements the application suite must define defaults and user access roles specific to their institution. A university student will only have access to enrolling in classes, checking schedules and paying tuition while a university professor will be presented with a different set of functionality that allows the upload of grades and review of detailed student information. A technical administrator is responsible for setting up defined defaults, such as department and cost center for a new user or group.

User Task: What are the different activities the users do with your application?

Table 2. User experience vs. developer perspective. The left column displays the user experience terminology prior to the research. The right column presents the developer terminology as a result of the research

User experience perspective of user task	Developer perspective of user task
Compare	N/A
Create New	Create
Monitor	Monitor/Track
Update	Update
Browse	Browse
N/A	Setup
N/A	Report
Seek Info	Search
Delete	Delete
Identify	N/A
Analyze	N/A
Model/Plan	N/A
Execute	N/A
Develop Strategies	N/A

Development had a much shorter list of user activities than user experience. Generally, the more abstract activities such as plan or analyze were not mentioned by development. The C.R.U.D. term was brought up a couple of times by development and is an acronym for Create, Retrieve, Update and Delete. Only *retrieve* wasn't specifically mentioned as an end user activity.

Data Structure: Do you think about the data structure when designing the user interface?

Table 3. User experience vs. developer perspective. The left column displays the user experience terminology prior to the research. The right column presents the developer terminology as a result of the research

User experience perspective of data structure	Developer perspective of data structure
Unstructured	N/A
List	N/A
Table	N/A
Hierarchy	Hierarchy

User experience had thought that development made decisions about the user interface *after* addressing key back-end challenges. This line of reasoning supposed that the interface was driven by prior decisions regarding the back-end implementation. As noted earlier, development did separate back-end decisions from the front-end. In fact, the front end prototype generally dictated back-end requirements. The one exception was for hierarchical data. This was the only data structure that had an effect on the interface.

Layout Structure

Table 4. User experience vs. developer perspective. The left column displays the user experience terminology prior to the research. The right column presents the developer terminology as a result of the research

User experience perspective of layout structure	Developer perspective of layout structure
Application	N/A
Process	Flow
Page	Page
Component	Component

There was no specific question to layout structure but we actively listened for references to design scope. Most changes to the interface were limited to page-level design. While user experience often focused on user flows, development framed the same flow as several incremental changes to existing pages.

4. Summary

The development designer viewed the design process as the most laborious, difficult and ill-defined part of the development cycle. Since there were no agreed upon criteria for choosing a design, contrasting points of view often surfaced that led to a longer development cycle. In most cases, user experience designers were not available during the design phase due to limited resources. In order to get through the front-end definition, developers became designers and used a variety of strategies to make design decisions.

The largest single factor in making a design decision was consistency with the current application. Long-tenured developers and product managers often provided guidance to junior developers on reapplying existing designs for new features. Development teams rarely explored design consultation from other groups. Developers and product managers noted that having too many people involved created too many opinions. Due to development's focus on incremental changes to an application, major innovation for the interface must be initiated from user experience.

Understanding the development designer perspective helped inform the design of the decision support tool user experience is planning to build. However, the development designer perspective terminology should not be used exclusively. We believe that a blending of user experience and developer terminologies will work best. Some user experience facet attributes, such as *compare* for user task, should be surfaced in the support tool interface since these terms may prod development to consider important user-centered design concepts. In the end, junior members of the development organization may benefit the most from investigating patterns with the decision support tool since they are not overly influenced by legacy designs.

References

1. Alexander, C. A., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., and Angel, S. A.: A Pattern Language: Towns, Buildings, Construction. Oxford University Press, New York (1977)

2. Gamma, E., Helm, R., Johnson, R., and Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, Mass (1995)

3. Tidwell, J.: Designing Interfaces: Patterns for Effective Interaction Design. O'Reilly, Sebastopol, CA (2005)

4. Yahoo! Design Pattern Web site: Retrieved May 8, 2006, from http://developer.yahoo.com/ypatterns/index.php (2006)

5. Turban, E.: Decision Support Systems and Expert Systems: Management Support Systems. Prentice Hall, Englewood Cliffs, NJ (1995)

6. Miller, C. A.: Bridging the Information Transfer Gap: Measuring Goodness of Information Fit. Journal of Visual Languages and Computing. (1999) 10(5), 523-558

About the authors

Brent White is a senior interaction designer within the supply chain management user experience team at Oracle Corporation. His research interests include software interface design patterns and decision support systems. He is a Masters of Science candidate in Human Factors and Ergonomics at San Jose State University.

Lynn Rampoldi-Hnilo is a user experience manager at Oracle Corporation. She manages a team of interaction designers and usability engineers to improve the user experience and satisfaction with Oracle supply chain management products. Lynn completed her Ph.D. at Michigan State University with an emphasis on technology and cognition.