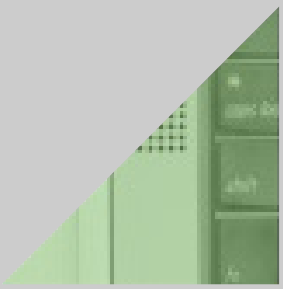
Best Team Practices for Designing User Strategy, User Experience, & User Interface for Web-delivered Applications



by Joseph Selbie & Michael Coombs

Study Advisor: Jakob Nielsen

tristream

TABLE OF CONTENTS

The Table of Contents is hyper-linked!

tristream

People always ask me to tell them how their UI should look. For some design problems, this is easy. For example, we know that users want a search feature on almost all websites, and we also know that they look for *"Search"* by scanning the homepage (or other Web pages) for *"a box where I can type."* It follows that the primary search UI should be a box directly on the homepage—not, for example, a link to a separate search page.

Even more complex problems sometimes have well-known solutions. For example, research into the behavior and needs of individual investors and investment professionals has identified 65 design guidelines for the investor relations sections of corporate websites (see http://www.nngroup.com/reports/ir). The reason it's possible to provide such detailed advice for IR pages is that investors have pretty much the same needs, no matter whether they are on company A's site or company B's site. In fact, often the users' goal is to determine whether A or B is the better investment for them.

We can even provide usability guidelines for some enterprise applications, such as the employee directory feature on an intranet. Again, the use of such directories is much the same from one company to the next, even though there are certainly differences based on factors like the size of the company and whether it's multi-national and/or multilingual.

The impatient reader will now be saying, *"Okay, what about the design of my application? What are the sixty-five things I need to include in my UI?"* Sadly, this is not so easy to answer. By definition, enterprise applications exist to provide specific functionality to solve specific business problems for specific users. The features needed, and their look and feel vary, depending on the characteristics of the problem and the users.

Even though there's no pre-defined cookie-cutter UI to solve your design problem, it's possible to provide a process to define a user experience that will work well for your users and provide huge productivity gains and other business value. As it turns out, even though the specific design differs by project, the steps needed to achieve good results are much the same across many projects. That's where the current report comes in handy: It has collected best practices for the process across multiple projects.

As you read the report, you will probably find yourself in agreement with much of the advice. It's good advice. It works. It results in better design. I caution you against believing that simply because the individual best practices are obviously good that they are therefore also obvious. Plenty of experience shows that they are not obvious—or at least that they are often not followed.

As you read the report, I recommend that you scrutinize and review your own team practices with a critical eye. For each recommended approach, is thus truly what you do? Is it just what you would have liked to do, but you slipped somewhere and started to run the project in a more traditional way? The difference between doing it right and doing it wrong is often subtle. Take, for example, the question of understanding user needs. Almost all development managers will claim that they make an effort to understand users before their team starts coding away. And yet many—if not most—IT projects fail because of a mismatch between user needs and what's delivered.

The point is that it's not enough to want to understand the users' needs. There are specific methodologies that need to be employed systematically to make sure that this happens to a sufficient extent. Same for each of the other issues discussed in this report.

In some ways this report describes an idealized process. There are probably very few teams that do everything right and follow every last recommendation, but there is no doubt that those teams that follow the majority of the recommendations tend to be more successful than those teams that ignore these best practices. If I were to make one meta-recommendation, it is that you try to implement at least some of the best practices on your current project. If you delay until you can run the perfect project, it will never happen. Do some now, and then do more on your next project.

— Jakob Nielsen, Ph.D.

The Tristream study team researched, evaluated, and defined the best practices of development teams that result in highly usable and highly productive user strategies, user experience, and user interfaces for web-delivered applications. Applications that were developed using a significant number of these best practices resulted in higher user satisfaction, ease-of-use, productivity, accuracy, efficiency, customer service satisfaction, reduced employee needs, reduced training costs, reduced help desk time, and bottom-line ROI.

Differentiating Practices

Our goal was to identify the practices used by teams that "stood out" from the rest. We did not try to catalogue all the practices in use by development teams—even when those practices are essential to development, such as many project management practices. Instead, we looked for practices that clearly resulted in superior application development and which were absent from, or poorly executed by other teams.

The best practices we identified range from team structure to team member skill sets, from user strategy development techniques to wire-framing, from internal documentation to establishing ongoing user groups, from prioritizing business requirements to working with offshore teams. In all cases, the best practices we identified, like cream, rose to the top.

"Core" Teams

In this, the first version of our best practices study, we concentrated on "core team" practices. Core teams vary in size from 5 to 15 people. The core team is hands on and works directly with a specific application, or portion of an application, and is responsible for the development of all the screens required, down to the last pixel. Frequently the "core team" hands off the programming to a larger group, often off-shored.

There are frequently multiple core teams working on the same application in large application development projects. In the next version of this report, we will address the best practices for managing and supporting multiple teams working on the same application. However, in this report we do not have sufficient information to address the coordination of multiple teams other than with an observation here and an observation there.

However, we will say without reservation that it is the core team that really insures great application development. All the coordination in the world cannot offset poor work in the core team—and vice versa, a great core team can even rise above some of the problems created by poor coordination.

So whether your company has an IT department under a hundred people, or in the thousands of people, you will find the best practices identified and described in this report to be very applicable to your core teams.

Off-shoring

Many of the best practice descriptions address specific ways to work well with off-shored elements of the development process, particularly the programming team. Off-shoring appears to be here to stay and the best teams have adapted to the opportunities and problems presented by an offshore team.

Challenges and solutions center on communication and cultural understanding, which are sometimes intermixed. Offshore programming teams in India, for example, are not comfortable with the directness of the typical

American conversational style, and will often take a passive role in the process as a result. The best teams make an effort to draw out their counterparts' good solutions, often sending team members to India on a regular basis.

On the other hand, offshore teams need to send key Team Leaders to their client facilities on a regular or long-term basis. Only then can the offshore team gain a real feel for the users and business needs around which the application is being developed.

Thorough communication is essential. Offshore teams generally need a more specific design specification than an internal programming team sharing the same facilities. This report addresses many best practices that alleviate many of the problems associated with off-shoring, including a special section, "Tips on Outsourcing."

New Applications and New Releases

Most of the best practices that have been identified in this report best apply to the creation of new applications and to the process of developing major new releases of existing applications. We focused on how the best teams applied their methods to applications where there were as few pre-existing limitations as possible. A clean-slate start, or a major new release (new version) of an application affords the maximum opportunity to implement greater usability and productivity.

However, the best practices we identified are also applicable to making incremental changes between major releases by recognizing pre-existing limitations and by a common-sense application of these best practices. The same principles apply, but typically a team isn't able to go as deeply into changing user strategy or the user experience as they might like to do when involved in incremental change. When making incremental changes, teams typically need to preserve the overall user experience conventions that are already in use throughout the rest of the application, otherwise the user will have an inconsistent and confusing experience.

Differentiating Between User Strategy, User Experience, and User Interface

The application development industry tends to treat user experience development (UE) and user interface development (UI) as the same process, sometimes using the acronym UE/UI. We have found that to be emblematic of many of the problems we routinely encounter—teams do not recognize or make distinctions between what should be separate phases or separate processes in the development process. As a result, teams often develop user strategy, user experience, and user interface elements out of sequence—or in the case of user strategy—not at all.

There is an old story (told with many variations) of a science professor filling up a glass jar in front of his students without any explanation. First he fills the jar with two or three large rocks. Then he adds as many medium-sized rocks as he can fit around the big rocks, then he fills up all the remaining space he can with fine rocks and sand, then finally he adds water right up to the very brim. Then he asks his students, "What did you learn from this experiment?"

Many students offer answers such as, *"You can always get more in the jar,"* or more metaphorically, *"You can always do more with your time."*

Finally the professor, acknowledging the truth of what his students offered, says, *"All that may be true, but the most important thing you learned was that you have to get the big rocks in the jar first."*

Using this analogy, user strategy corresponds to the big rocks. User strategy typically addresses big picture considerations such as, where does this application fit in with the rest of the company's information systems? Is it part of a portal? Does it serve vendors and customers as well as internal employees? What is the basic "flow" of the application? Does it mirror internal divisions, such as sales, accounting, etc.? Or does it mirror workflow, such as initial call, proposal creation, order confirmation, fulfillment, follow-up? Is the application primarily for new users? Occasional users? Power users? Will the application have more than one "track," or have significant user-preference settings?

If these overarching user strategy decisions are not consciously made in the beginning of the development process, user strategy will be determined almost randomly by a series of other decisions—and it is rarely clear, usable, or productive as a result. As the old saying goes, "If you don't know where you're going, any road will get you there."

The best teams address user strategy first and make it a distinct step in the process. Your user strategy will probably need to be adjusted as the development process unfolds, and as new needs or limitations emerge, but the core strategy should be strong enough to survive such adjustments without losing clarity.

User experience development flows from user strategy and corresponds to the medium-sized rocks. Once overall user strategy has been decided upon, the best teams address themselves to understanding their users' task and workflow. The best applications make the users feel as if the designers knew exactly what they want to do next. In addition, the best applications recognize distinct user types and allow them to perform their tasks without any confusion. New users, or occasional users, will not have an agreeable user experience in an application designed for everyday power users, and vice versa.

There are no hard and fast "best user experience" conventions. If a team were to put all the best conventions together in one application, it would be a confusing mess. The "best user experience" is one that is clearly consistent and matched well with the specific needs of each user type.

Great user experience design is born of an intimate knowledge of business needs, the users' needs, and a robust, iterative wireframe design process. We did not find a single excellent, or even good, application that was not first designed visually. A user's experience is primarily visual. What he or she sees on the screen is the basis of his or her experience.

User interface design flows from user experience design and corresponds to the sand and water that go into the jar last. The user interface is made up of the visual elements seen on the screen—colors, shapes, banners, headers, modules, typography, icons, buttons, and controls. The user interface "clothes" the user experience in an easy-to-see, easy-on-the-eye ergonomically developed toolkit of interface elements.

The basic development flow—user strategy, to user experience, to user interface—insures that the most important decisions get made first and then cascade down to the smallest decisions.

Early in the Development Lifecycle

This study focused on the elements of the development lifecycle that had the most influence on creating highly usable and highly productive applications. We found, not surprisingly, that the initial development stages have the

most to do with creating great applications: determining business requirements, gathering user requirements, designing user strategy, then user experience, then the user interface elements and, throughout the process, good communication, and documentation.

We did not spend significant time studying acceptance-testing, IT implementation, QA, user-training, initial launch, and other very important steps to getting an application developed, because we found that these activities, falling late in the process as they do, had a diminishing impact on the final quality of the application.

The most important work—as far as usability and productivity gains are concerned—happens in the early stages of the development process. If you don't get things right early, it is very hard to do much about them later in the process.

Across All Types of Applications

The best practices we identified and describe in this report work for any web-delivered application. We included both publicly accessible Web sites, employee-only "intranet," and business-to-business applications in our study. The best practices would, however, be applicable to any process involving a screen, whether a full-sized computer monitor, or a small, handheld screen.

The applications we studied are all on the critical path of each company's service delivery or internal operational processes or, where directly accessible to consumers, are central to the company's revenue generation. The majority of applications under study are internal and/or B2B applications and therefore our study results are strongly applicable to teams engaged in the development of these types of applications.

However, the development teams that were engaged in the development of B2C applications, and who employed these best practices, also engendered high user satisfaction and productivity.

Company investment in the applications was significant to extremely significant. The companies that participated in the study ranged in size from small ($20 million annual revenues) to global (Nokia with nearly $40 billion annual revenues). The applications under study ranged from small, single-purpose applications, to enterprise-wide applications with dozens of user types and thousands of users. Development team size ranged from 5 to 100 (depending on the size of the programming team). Development budgets ranged from hundreds of thousands, to tens of millions of dollars. Development time frames ranged from a few months to as many as five years.

Despite this wide range of company types, application types, sizes, and budgets, development team sizes and development time frames, very clear and consistent best practices emerged that cut across all these factors. Tristream found clear and universally effective best practices for gathering user input, gathering business requirements, developing user strategy, user experience, and the user interface, team dynamics and composition, and documentation that any development team can take advantage of regardless of team size or project scope.

Across All Types of Tools

Our evaluation does not address the strengths or weaknesses of any particular development processes (such as RUP, GPLC, XP), support applications (such as Visio, Rational Rose), or development tools (such as UML). The decisions which drive the selection or deployment of processes, support applications, and tools relate more to the

budgets, established culture, and preferences of the individual companies. By themselves, they do not insure, or work against, creating a great application.

The best practices Tristream identified cut across all processes, applications, and tools.

Best Practices Contrasted against "Normal" Practices

In the detailed descriptions of the best practices, you will find that we also include the "Normal Practices" that we encountered as well. We found that many teams considered themselves to be following the best practices we shared with them as the study concluded—when in fact we knew that they were not fully following the best practices. Perhaps they were short-cutting some of the key aspects of the best practices, or perhaps they had been doing it one way for so long that they could no longer truly see the fine points of what they were doing.

We therefore added descriptions of "Normal Practices" which will allow development teams to contrast themselves against actual best practices. The differences are sometimes subtle and therefore we tried to provide as many quotes and anecdotes as possible to illustrate both the best practices and the normal practices.

Bottom Line

Where the best practices identified were used, application usability and productivity were clearly superior; where they were omitted, application usability and productivity were clearly inferior.

BACKGROUND

Enterprise computing is in transition from the client/server model to a web-based delivery model. Many companies have made significant steps in moving to web-delivered applications, thousands more are in process, or are just beginning. The transition appears to be inevitable, as surveys indicate almost all IT departments in major corporations intend to make the transition in order to take advantage of the flexibility and freedom of web-delivered applications—whether it's now or at some time in the future.

While underlying technologies, such as data bases, programming environments (such as .net), offer solutions to many of the technical problems encountered in the transition to web-delivered applications, the processes for the development of user strategies, user experience (UE), and user interface (UI) have not kept pace. The result of these practices lagging behind the technology potential is that most applications take neither full advantage of new technology, nor provide high user satisfaction. Both results prevent companies from realizing the full ROI of their technology investment—most applications therefore are neither user-friendly, nor highly productive.

The transition from the client/server model to a web-delivered model for applications has brought together two independently evolved methodologies for developing applications. The client/server model, the backbone of corporate computing for decades, has been supported by rich and stable development methodologies, which have had decades to mature. The methodologies that have evolved with the Web are newer and less well-defined. Developing web-delivered applications requires that development teams merge these two processes.

While the two development processes (for client/server and Web development) do not differ greatly, there are important differences, particularly where user strategy and user interface development are concerned. Traditional client/server development projects had the benefit (or, at the same time, the obstacle) of an already developed user interface, which in turn would drive the potentials for user strategies. Most programming environments, which were designed to develop client/server applications, came with well-developed UI components.

In the web-delivered application arena, the user interface cannot be treated as a given, as it often was in the client/server application development process. The wide variety of browsers that can be (or need to be) supported, and the continuing evolution of display code and browser capabilities, make user interface development for web-delivered applications more challenging—and more of an opportunity—to deliver highly usable interfaces.

On the other hand, user strategy and user experience development, while partly driven by the capabilities of the user interface (UI), remains the core challenge of either development methodology—and neither methodology insures good results. It is here that we found the greatest weakness among the teams evaluated, regardless of the overall methodology their team employed for the development process.

Our thanks to the participating companies listed below. They provided generous and unfettered access to their applications, their users, and their development teams. Without their cooperation this study would not have been possible.

The Tristream study team interviewed from six to eight users from each primary user group that use the applications in order to evaluate their usability and productivity. Tristream then ranked each application according to user perceptions of usability and productivity. Additionally, all members of the Tristream team, after familiarizing themselves with the applications under study, ranked the applications by the same criteria to add a greater measure of objectivity to the results.

The user interviews and ranking process established the comparative success of each application in terms of satisfying user needs and meeting work goals. The ranking then provided a baseline of success for evaluating the practices of the teams that developed the applications.

We then evaluated the development practices, team dynamics, and team composition of each development team. The development teams interviewed were internal, though in some cases partial outsourcing was used to help with the development of the user experience strategy, the user interface design, or code development.

KPMG (Global)—2 applications, 2 development teams

KPMG is one of the leading providers of audit, tax, and advisory services with over $13.5 billion dollars in annual revenues. KPMG member firms respond to their clients' complex business challenges with a global approach to services that spans industry sectors and national boundaries. Considered one of the "big four" global accounting firms, along with PricewaterhouseCoopers, Ernst & Young, and Deloitte Touche Tohmatsu.

Application #1 - An online tax planner and travel calendar for expatriate employees (employees who are working in a country other than their own). An end user online application requiring a high degree of usability (expatriates only use it a few times a year) yet rigorous in its capability to capture all required information accurately in order to assist expatriates with filing their taxes in two or more countries.

Development Team #1 - Small to medium-sized (5 to 10 people, depending on development phase). User strategy, user experience, and the user interface development were partially outsourced.

Application #2 - A multi-tiered software application that enables large multinational organizations to efficiently manage and administer various aspects of their international assignee programs. It serves companies that run their global assignment program from one or more decentralized locations. It allows companies to structure compensation packages, determine tax positions, and document policy changes. It also provides demographic information, calculates allowances, and processes non-payroll expenses, reducing the effort needed for complex accounting and reporting requirements.

Development Team #2 - Medium-sized (10 to 15, people depending on development phase). User interface development was partially outsourced.

Nokia (Global)—2 applications, 2 development teams

Nokia is a world leader in mobile communications, with nearly $40 billion in annual revenues, driving the growth and sustainability of the broader mobility industry. Nokia connects people to each other and the information that matters to them with easy-to-use and innovative products like mobile phones, devices, and solutions for imaging, games, media, and businesses. Nokia provides equipment, solutions, and services for network operators and corporations. Nokia is a broadly held company with listings on four major exchanges.

Application #1 - A human capital management application, it serves nearly 60,000 Nokia employees to help them set and track career development objectives and evaluate performance. It is designed to help Nokia methodically build employee competence in the direction of Nokia's larger objectives.

Development Team #1 - Small to medium-sized (5 to10 people, depending on development phase). All development is done in-house.

Application #2 - This application allows access and scheduling for all of Nokia's conference rooms and conference room services, such as computer, audio and visual equipment, catering, and visitor management throughout all of Nokia's global facilities. Single-purposed, yet widely used, employees can schedule conference facilities from anywhere for any location.

Development Team #2 - Small to medium-sized (5 to10 people, depending on development phase). All development is done in-house.

Anonymous 1 (United States)—1 application, 1 development team

This company, which has chosen to remain anonymous, is a leading provider of customer service and contact center software for in-house or on-demand deployment with nearly $20 million dollars in annual revenue. It helps organizations achieve and sustain customer service excellence by transforming traditional call centers into multi-channel customer interaction hubs, and to extend their service-based competitive advantage.

Application #1 - A robust call center application, serving agents, administrators, and knowledge-base developers. Multi-channel by design, agents can process email, phone, and chat interactions through the same user tools. Able to be deployed for a single company or as a single application serving multiple companies.

Development Team #1 - Large (35 to 50 people in continuous development). All development is done in-house, although the company maintains a facility in India for their own programming group.

Islandsbanki (Iceland)—1 application, 1 development team

Islandsbanki's principal activity is the provision of retail banking services to both private customers and businesses. Other activities include asset management and financing, treasury and risk management, merchant banking and brokerage services, with a combined annual revenue of over $500 million and assets over $8 billion. It has five fully consolidated subsidiaries, of which three are located in Iceland, one in Norway, and one in Luxembourg.

The Group entered the Norwegian market with the acquisition of KredittBanken during 2004. In addition, when the Group acquired Sjova-Almennar tryggingar hf in 2003, it entered the insurance market.

Application #1 - A full-featured online banking system for consumer, corporate, and merchant banking, including account set up and management, bill paying, loan acquisition for homes and cars, automatic loan payment, and insurance and equities services.

Development Team #1 - Small to medium-sized (5 to 10 people, depending on development phase). All development is done in-house.

Anonymous 2 (Global)—2 applications, 2 development teams

This company, which has chosen to remain anonymous, is a business unit of a large U.S.-based company with nearly $20 billion in annual revenues. The company is an industry leader in global mobility and workforce development support to organizations worldwide. Clients include global and locally based organizations of all sizes and types. A broad base of services supports both managers and transferring employees with cost-effective, customized, mobility management services.

Application #1 - A real estate referral application providing a nationwide network of real estate brokers and agents with an extensive referral system which distributes and captures referrals between broker ages, allowing brokers and agents to track referrals, sales status, and commissions across all 50 states.

Development Team #1 - Small (5 people). The development is all done in-house.

Application #2 - A call center and customer service application which coordinates customer service and information processing for the core service delivery and revenue-generating functions of the company. The application enables call center capabilities with real time information retrieval across the world and acts as the central information processing hub for a globally dispersed 2000+ member customer service consultant and customer service specialist group.

Development Team #2 - Large (15 to 25 people, depending on development phase). User strategy, user experience, and user interface development were partially outsourced.

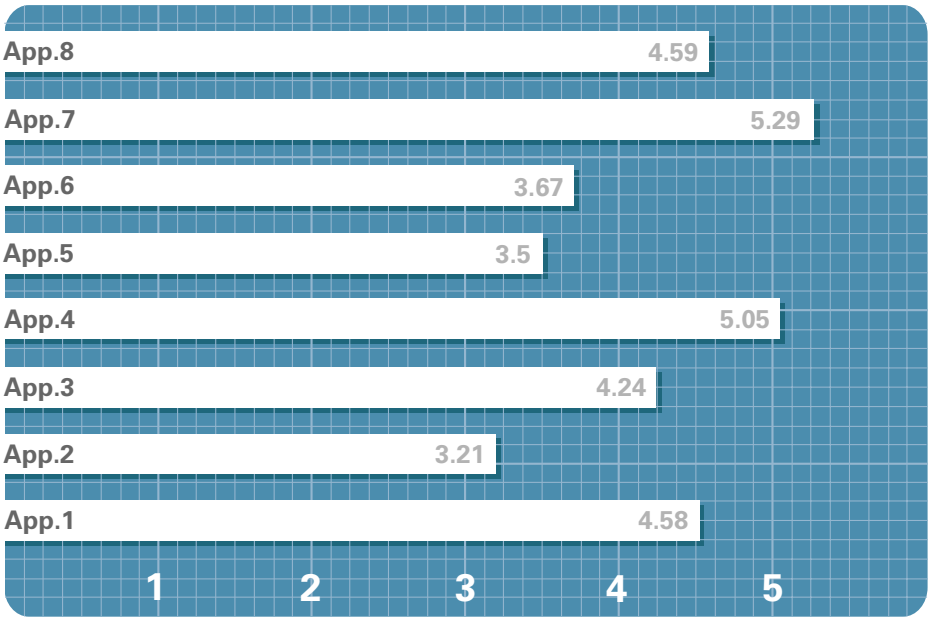
User Rankings

Each application was ranked by users on their satisfaction level in regard to various criteria. (The user interview questions, as well as more information on the study methodology, are available in the appendix.) Users were asked to rank the application on the following scale:

6 - Completely Satisfactory
5 - Mostly Satisfactory
4 - Somewhat Satisfactory
3 - Somewhat Unsatisfactory
2 - Mostly Unsatisfactory
1 - Completely Unsatisfactory
0 - Not Applicable

The applications, which have been kept anonymous, received the following average user rankings and comparative rank in comparison to the other applications:

	App. 1	App. 2	App. 3	App. 4	App. 5	App. 6	App. 7	App. 8
Application Rank by Users	4	8	5	2	7	6	1	3
User Averages	4.58	3.21	4.24	5.05	3.5	3.67	5.29	4.59



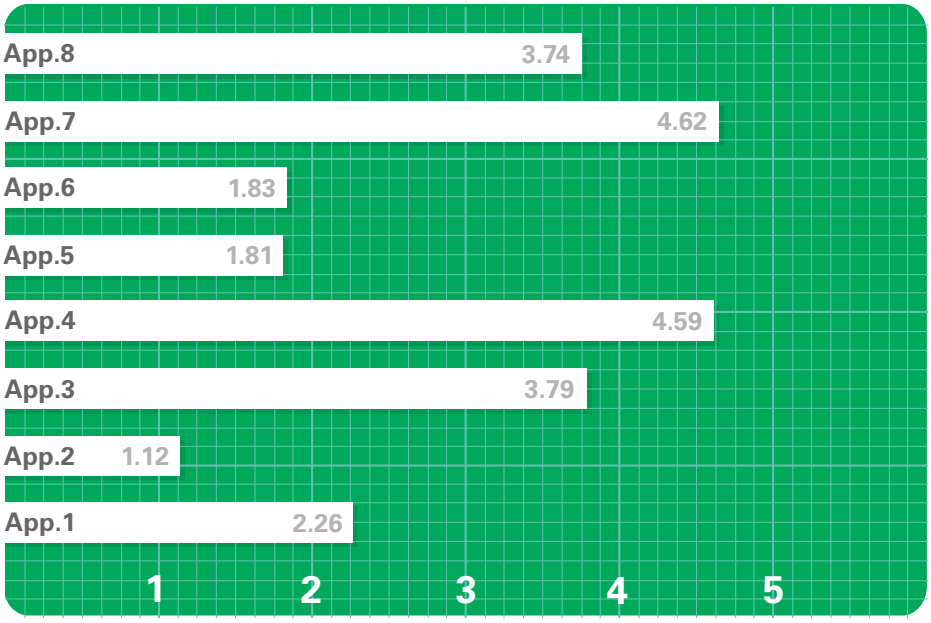
Team Rankings

Once the user interviews were concluded, and the best practices were identified, the Tristream study group ranked each team on how well they applied each of the 58 best practices we identified to the development of the same applications that the users were asked to rank. We ranked the teams on how well they applied best practices on the following scale:

6 - Exceptional
5 - Good
4 - Moderately Well
3 - Average
2 - Below Average
1 - Poor or Completely Missing

The teams received the following average rankings from Tristream and comparative rank in comparison to the other teams:

	App. 1	App. 2	App. 3	App. 4	App. 5	App. 6	App. 7	App. 8
Application Rank by Teams	5	8	3	2	7	6	1	4
Averages	2.26	1.12	3.79	4.59	1.81	1.83	4.62	3.74

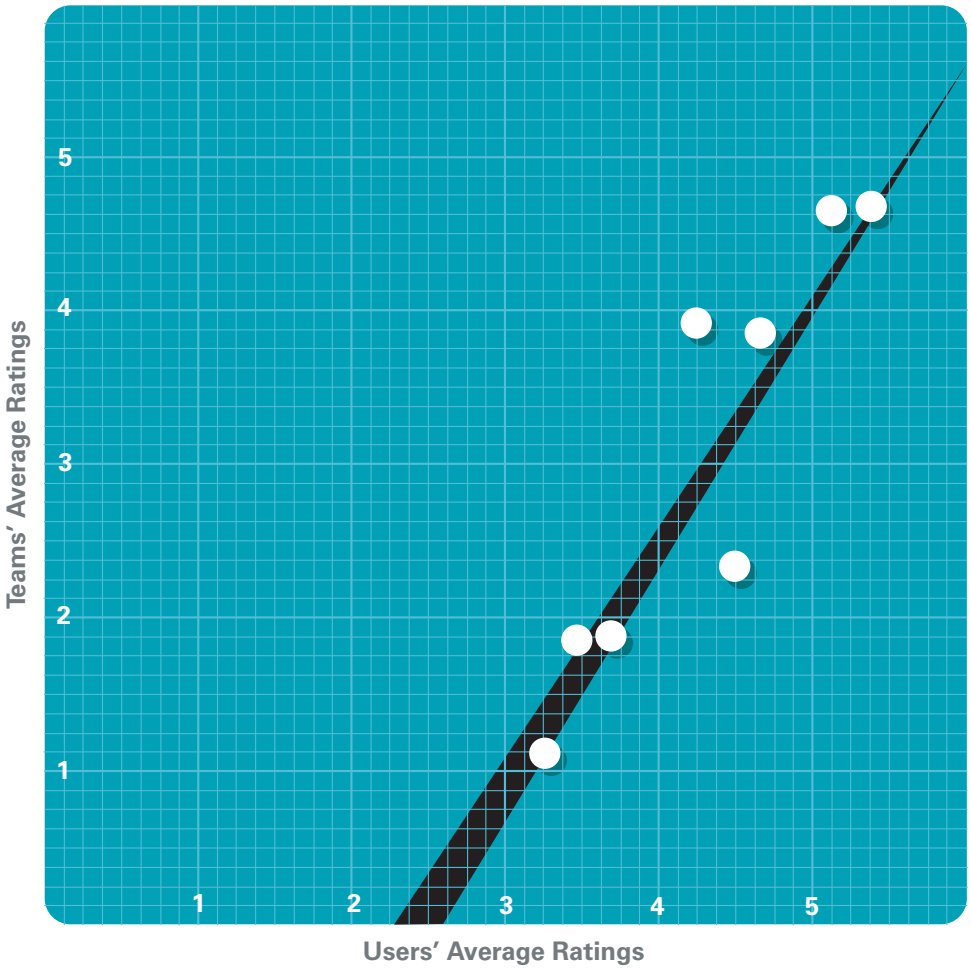


Comparison of User Rankings and Team Rankings

After completing the ranking process of how well the teams applied best practices, we then compared team rankings to user satisfaction rankings to see if there was a correlation. You will see in the charts and graphs below that there was nearly a 1:1 correlation for the best applications and best team use of best practices:

	App. 1	App. 2	App. 3	App. 4	App. 5	App. 6	App. 7	App. 8
Application Rank by Teams	5	8	3	2	7	6	1	4
Averages	2.26	1.12	3.79	4.59	1.81	1.83	4.62	3.74
Application Rank by Users	4	8	5	2	7	6	1	3
Averages	4.58	3.21	4.24	5.05	3.5	3.67	5.29	4.59

The following scatter plot chart further demonstrates the extremely high correlation between user satisfaction and the use of these best practices by development teams. In fact, we were very pleased to receive this quote from our study advisor Jakob Nielsen: "The correlation is r = .90, which is extremely high." (An r value of 1.0 would indicate an exact correlation.)



The Exceptions

We were pleased by the obvious and extremely high correlation of using best practices and achieving high user satisfaction. However, as you can see below, three applications in the middle of the rankings did not conform so neatly:

	App. 1	App. 2	**App. 3**	App. 4	App. 5	App. 6	App. 7	**App. 8**
Application Rank by Teams	**5**	8	**3**	2	7	6	1	**4**
Averages	2.26	1.12	3.79	4.59	1.81	1.83	4.62	3.74
Application Rank by Users	**4**	8	**5**	2	7	6	1	**3**
Averages	4.58	3.21	4.24	5.05	3.5	3.67	5.29	4.59

After looking through our rankings in detail, we saw several things that we believe explain the exceptions:

Application 8

You will notice that the average ranking score for the team that developed Application 3 (3.79) and the team that developed Application 8 (3.74) is so close as to be insignificant. Had the average team scores for the team that developed Application 8 been only .06 higher there would have been a one 1:1 correspondence between the team ranking (3) and the user ranking (3) for Application 8.

Application 3

The largest discrepancy is to be found in Application 3. The team received a high ranking from us (3), yet the actual application users ranked the application two places lower (5). In reviewing our notes we believe we have found several explanations.

The team that developed application 3 was the smallest of all the teams we studied. They were also all located together, which is easier for a small team to achieve, which made it easier for them to practice many of our best practices—such as achieving a holistic understanding of the application, maintaining fast and fluid communication, and rapid decision-making—thus getting high scores from us in these areas.

However, it was the first time that most of the team members had done a web-delivered application and it was a first version of the application. Many of the decisions made—which strongly informed the project, and which were supported by good user awareness—were nonetheless made by a relatively inexperienced group of people and they simply did not get them right.

From this we would readily conclude that best practices by themselves cannot replace the need for well-trained and experienced team members.

Application 1

On further study, we had to conclude the opposite regarding Application 1. It received a relatively poor ranking (5) from us regarding their use of best practices, yet received a higher ranking from its users (4). We believe that the explanation lies in the fact that this team has been working together on the same web-delivered application for some time, and are now up to version 3. The team also outsourced a significant portion of their user strategy, user experience, and user interface development. So we had to conclude that the greater experience and skills of their team members (combined with the outsourced work) compensated for their relatively poor utilization of best team practices.

Conclusion

Had we seen the exceptions, described above, at the top of the rankings, and at the bottom of the rankings, rather than simply in the middle ranks of the applications, we would have been very disappointed by our findings and would have had to reluctantly conclude that we missed vital best practices, or did not clearly enough identify the best practices we observed.

However, the unequivocal correspondence between the highest ranked applications by user satisfaction and the fullest utilization of best practices by the teams that developed them gives us great confidence in the value of the best practices identified in this study.

There is no substitute for experienced and talented team members, but combine talent and experience with these best practices, and you and your team will create excellent, user-centered, highly usable, and highly productive applications.

TEAM DYNAMICS

1. HOLISTIC UNDERSTANDING

The best teams achieve a holistic understanding of the application under development. By holistic we mean a thorough understanding by all core team members of all aspects of the application, including user needs, business requirements, user strategies, user experience, and user interface.

In the best teams the user is king. There is a mature appreciation of the importance of end user needs. The best teams do not let personal preferences, or over-familiarity with the application, to cause them to lose sight of the end users' perspectives.

In the best teams, all team members are aware of the entire development process, and as much as possible participate in the entire process, such as determining business requirements and gathering user needs (especially). They participate in the various stages of review (such as use cases, wireframes, design specifications, etc.), and most importantly, participate in all stages of user strategy and user experience development.

When teams achieve a holistic understanding, the resulting user strategies, user experience, and user interface that are developed are significantly more usable and productive. The development process is also more efficient: There are fewer conflicts regarding user strategy, far fewer revisions, and missteps, and an overall clarity that allows for the rapid working out of the myriad problems requiring solutions in keeping with the overall user strategy and user experience of the application.

As we were told by a team member: “No matter how good your documentation is, it can never be as good as having done it all and knowing it inside and out.”

Best Practices that Foster Holistic Understanding

a. Team members remain involved for the duration of major development phases of the application, such as initial development or major revisions.

- Even when a team member's particular skills are not yet required full-time on a project, the team member is still significantly involved in, or aware of, all phases of the development process.

> One Team Leader realized that the programmers assigned to the application development process had very little idea of the business goals for the application, or of the importance to the company of the eventual success of the application, and therefore felt little real connection with the project—it was just a bunch of code. To remedy this, the Team Leader took the programmers to various meetings with the actual end users and business leaders who needed the application—even to the point of renting them tuxedos and taking them to an awards banquet! The end result was a significantly deeper understanding of the business goals and user needs by the programmers, a significant increase in their commitment to the success of the application, and a dedication (often involving extra hours, extra revisions, and extra effort) to getting it right.

- Special effort is made to include the programmers in the front-end process.

  > In-house programming teams choose a lead developer to participate, as much as possible, given the other demands on their skills, in the front-end design process. The participation of the lead developer insures, more than any other single factor, that the end results of the front-end planning process are programmable and take maximum advantage of IT infrastructure capabilities and opportunities.

  > Successful teams in companies that choose to outsource and/or offshore their programming insist on the onsite presence of at least one leader from the programming team throughout the front-end development process.

  > Many outsourcing firms, with off-shored staff and facilities, have an in-country staff that manages sales and manages projects at a high level. Having regular onsite visits from in-country staff is not sufficient to insure inclusion of the programmers in the process. The in-country staff does not end up doing the actual programming work. The best teams insist on the regular presence of an onsite Programming Lead from the actual group of programmers assigned to the project.

b. The Team Leader remains with the application throughout its lifecycle. The Team Leader, more than any other role, insures continuity and transference of the holistic view of the application to new team members.

c. Front-end team members (Business Analysts, Business Representatives, stakeholders, etc.) stay closely involved in a frequent and systematic review process once the design specification has been "handed off" to the programming team. This insures that core business goals, and primary user experience solutions do not inadvertently get lost in the programming and implementation phase.

d. Team members work full-time on the application, allowing them to maintain a steady focus on the project to create new solutions and creatively solve problems.

- *"We actually have dedicated, assigned people in roles on the team. You can't be successful with too much multitasking. I think we'll have a higher quality, on-budget, and on-time project with this dedicated team."*

e. Team members are cross-disciplinary. Many team members are grounded by experience in more than one skill set necessary to the development process.

f. Documents generated by various team members are presented to the entire team and then discussed. Everyone is made aware of the implications of new information, changing business goals, new user needs, and new IT challenges, and their impact is discussed together in group meetings. Document size is kept to a minimum to insure that they will be read and referenced. Documents are freely available to all team members through a single document storage area.

- See Documentation for more on this.

Normal Practices

a. Team members are brought into the process when their skill set is required. Team members are phased in with little or no time to be brought up to speed on prior phases of development. This is particularly true of programming teams who often come into the process when use cases or design specifications have been completed.

- It is here that we most often saw, or were told about, significant problems with outsourced programming groups, especially those that were off-shored. Typically, with an in-house programming team, it is possible to hand off a design specification that is not absolutely determined down to the last pixel, because the in-house team has a feel for prior applications, they have at least some awareness of the users, and decisions they make in implementation are informed by that knowledge. Teams that are outsourced, and to whom a design specification is merely handed, have no feel whatsoever for the project or the users. As a result, the decisions they make in implementation end up seeming almost random, are often inconsistent, and when pressed, they have no good rationale for their decisions. Unless your team is prepared to give an outsourced team an airtight specification (time consuming, cumbersome, and not absolutely possible) having Programming Leads from the outsourcing team significantly present through the planning process should be your requirement.

b. Team members are "silo-ed" in their area of specialization. They use their own training and their own "best practice" methodology irrespective of other team practices. Documentation and methodology often do not mesh as a result.

c. Programming leads wait for the front-end process to be completed and handed off, and then make their own decisions regarding implementation based on infrastructure limitations or internal programming standards known only to them or their team.

- We don't want to seem as if we are picking on programmers, but we did often encounter a bit of a disdain for experience design from many of them. "Just pretty pictures" was the frequently made comment—half in jest, but half serious as well. The programming staff often does not take the wireframes—representations of actual user screens—seriously, and assume they represent suggested directions rather than pixel-perfect representations of how the screens should look in the end. This lack of respect or understanding on the part of the programmers was usually a result of company culture more than individuals holding strong opinions. Successful teams had mutual respect between the roles and functions of the design and programming team members.

d. Business analysts, and other business representatives, drop out of the development process once the handoff to the programming group has been made, and only see the application again at a working "beta" stage—when significant change is no longer feasible.

e. Specialists, such as Business Analysts, work on several applications in succession, often with overlapping schedules, which tends to mitigate against a clear holistic view of one application.

f. Programmers see their work as "just code." *"Give me the specifications and leave me alone."* However, any specification, no matter how good, requires interpretation and understanding in a larger context. Without the understanding of the larger context, programmers make decisions arbitrarily or by personal preference regarding user experience.

g. Project managers are heavily oriented toward the programming phase (often current or former programmers themselves) and wait out the user strategy, user experience, and user interface development process until they can "get to work."

2. FLUID AND FAST COMMUNICATIONS AND DECISION-MAKING

The best teams achieve a culture of fluid and fast communications. Application development is highly iterative. Teams that communicate and make decisions quickly are able to process more iterations than those teams bound by more rigid communication processes, and as a result they can achieve greater improvement by processing even more iterations.

In the best teams, decisions are made as quickly as possible to maintain forward momentum. Past decisions are changed when new information, or better ideas come along—but only when they add significant value to the application. A good team, and especially a good Team Leader, knows the difference between "feature creep" (the tendency for new ideas to come along endlessly and push off completion) and essential improvements. However, a good team is not afraid of change, and will embrace new iterations, even if the change requires significant rework, if they can see a superior outcome as a result.

This is where the Team Leaders really prove their worth!

Best Practices for Achieving Fluid and Fast Communication and Decision-Making

a. Team members are located together as much as possible. This facilitates informal conversation, brainstorming, and communication, the opportunity for which is lost when teams only meet for specific goals at specific times. A good team lead will relocate team members that work in one facility into one location (when the project duration is sufficiently long to justify it).

"We found out from experience that it was very important to have our main business owner and our main technical owner working closely together, as in daily, as in sitting next to each other," one Team Leader told us.

However, it is frequently impossible for all team members to be located together:

- The best teams overcome this by flying team members in to participate in group processes (such as user research, work and task flow, or wireframing workshops) to maximize the amount of time team members actually spend together.
    a. *This observation probably enters the realm of psychology rather than application design, but as we observe teams working together as a group*

in the same location for several days, there is an indefinable "coming together in understanding" that takes place, which no amount of conference calls can replicate.

- Team members are encouraged to call one another informally without appointment or a previously arranged meeting, simply to discuss ideas, problems, and decisions that have been made.
- Outsourcing firms involved in the project, particularly off-shored programming firms, are required by contract to provide an onsite representative from the beginning of the project, or to be onsite at frequent intervals. The onsite representative is a senior team member for the outsourcing company.

- Team Leaders facilitate communication in dispersed teams through various means:
  - Conducting regular status and scheduling conference calls.
  - Creating an online war room to which is posted continuously updated information regarding deadlines, recent deliverables, upcoming milestones, killer stuff the team has created, etc.
  - Arranging for team members to share deliverables with the entire team.
  - Pointing out team and individual successes.
  - Communicating forward momentum, progress, and excitement.
  - Insuring document availability to everyone through a centralized document vault.

b. The Team Leader's primary responsibility is to keep the flow of decisions happening as quickly as possible. Team members need to receive fast, accurate, and reliable decisions to inform their ongoing work. Decision cycles are kept to days, even hours—not weeks.

- The best Team Leaders take the attitude that he or she is there to support the team members by providing timely information and decisions, so team members can move forward with their work.
- We have observed few things more important to team morale than a strong forward flow of momentum. Nothing kills momentum faster than long periods (which in application development time can mean hours to days) waiting for decisions. And nothing is worse than decisions that have to be reversed, are unclear, or confusing. It is better that a Team Leader significantly reduce the amount of time spent in hands-on work than to have the decision process go off the rails. Team members "get in a flow." The more a Team Leader keeps them in the flow the better the end result.
  - **>** We interviewed one team that had no enthusiasm for their project at all. In fact, they were disparaging and sarcastic about the application. It turned out they had gone through so many significant changes of scope and direction that they just wanted the project to be over.

c. Management, above the level of the team lead, respects the need for fast decision-making and manages application projects by setting clear goals at the beginning of the project and allowing the team lead maximum freedom within those goals. The best team leads sort out issues with those in

higher decision-making roles quickly and definitively. The skills and practices that allow good team leads to accomplish this are beyond the scope of this report—but it is worth noting its critical importance to successful teams.

- Good upper management and good company culture form a necessary backdrop for all highly successful application development projects. The companies we saw that prided themselves on fast, clear decision-making always helped the development teams be more successful.

d. The best teams have an atmosphere of "we're all equals," which promotes cross-discipline thinking, free communication, and an awareness of the whole project.

e. Clear, up-to-date documentation is maintained and accessible to all team members (see Documentation), and team members know and trust that the documents are reliable and up to date.

Normal Practices

a. Team members are physically spread out (even within the same facility) or are physically located with only small sub-groups of the team. Often individual team members are geographically dispersed, even internationally dispersed, and communicate solely via conference [con] calls.

b. Portions of the project are outsourced (frequently off-shored as well—particularly the programming) and significant communication with the outsourcing staff does not take place until handoff. Outsourced staff has no involvement in earlier phases of the project and therefore no appreciation for the core paradigms and concerns that have informed the project to that point.

c. Small subgroups, such as the programmers, Business Analysts, or user research staff, work together as small groups, but do not come together as a larger cohesive group with all the other team members. This is particularly true of programming groups.

d. Key decision-makers are often very busy and are unable to review/approve project deliverables (business requirements, feature specifications, user research, wireframes, use cases, visual designs, design specs, etc.) as they are completed, and take a week or two to give approval or make a decision. This results in long stalls in project momentum and cuts down significantly on the number of iterations possible.

e. Key decision-makers do not have sufficient time to familiarize themselves with project deliverables requiring approvals or decisions and therefore make inadequate decisions as a result.

- This has less to do with the team than it does with company culture, but a collection of decisions from business that contain contradictions or muddy thinking will bedevil a project right on through to completion. It can be compared to building a house with the foundation out of square. Everything that is added above the foundation has to be tweaked and forced into conforming with the out-of-square foundation, and follows the builders right up through the roof. If the final application looks a little "out of square" when it is completed, it is some times no fault of the application team.

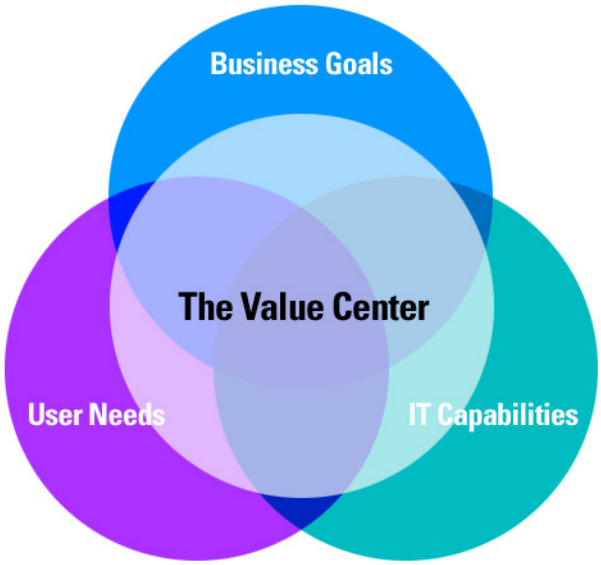
f. Project documents are shared as email attachments or notification via email that documents are available at certain locations. Documents are frequently dauntingly large and do not communicate to other members of the team without explanation. Teams are often unclear on where the most recent documents are to be found, and unclear as to which documents are in fact up to date and rely on the original authors of the documents to send them new versions.

We can't tell you how many meetings we've attended where a conversation goes something like this:

"But that's not what the specification says—"
"Yes, it does; didn't you get the last version sent to you?"
"Do you mean the one Bob sent last week?"
"No, not that one, I sent it out on Tuesday. Didn't you get it? It was in the email with the last wireframes we revised."
"Oh, that one. I didn't have time to go through the whole document . . ."

3. FIND THE VALUE CENTER: OPTIMIZATION OF BUSINESS GOALS, USER REQUIREMENTS, AND IT CAPABILITIES

The highest ROI for an application requires the optimum balance of three major factors; under emphasize or over-emphasize any one of them, and the application does not deliver its maximum value to the organization. Therefore, team composition and processes must reflect a balance of these three factors:



a. Business Goals

- Developing a new application, or major new release of an existing application, is an enormous opportunity for the business, not to just fix problems (often the first impetus for the new development project), but to improve or streamline business processes, and in parallel, to create an application that supports the improved business processes in a highly productive manner.
- The Stakeholder Champion, Business Analyst(s), User Representative(s), and Team Leader are crucial to making this happen.

b. IT Capabilities

- Maximizing IT capabilities requires a team that will push themselves for innovative solutions in balance with the long-term need for efficient maintenance of the code and database.
- The Lead Programmer(s), UI Developer, Project Manager, and Team Leader are crucial to making this happen.

c. User Needs

- Creating highly usable and highly productive applications is possible only when user needs are carefully considered. An application can be highly productive when user work and task flow are thoroughly understood. An application can be highly usable when the user strategy, user experience, and user interface are based on user research and testing.
- The User Representative(s), Business Analyst(s), UI Developer, and Team Leader are crucial to making this happen.

Normal Practices

a. Business Goals

- There is no Stakeholder Champion at all. Requests for business process change, or for improvements in an existing application, percolate up from operational manager's change requests, user complaints, or recurring help desk problems, and go to the Business Analysts or Programmers who then put this forward to "management."
  > Typically there is no process time applied to sorting through this disparate collection of input. It is, and remains, a jumble of business and user needs and the application upon which it is based will also feel like a jumble. There is a touching, but misplaced, faith that if all the functional requirements for the application are met it will somehow satisfy the users.
- Where there is a Stakeholder Champion they are often more stakeholder than champion and are most interested in getting "problems" fixed in the existing application in a timely manner.
  > This is often no fault of the stakeholder. The company has not allowed enough time to the stakeholder to give the project the attention it deserves. More often than not, responsibility for overseeing the project (from a business point of view) is merely added to the stakeholder's current list of responsibilities. Smart companies realize the need to allocate more time to their stakeholders, even in some cases removing them from most or all of their usual responsibilities until the application is completed.
- The Business Analyst, while aware of opportunities for business process improvements, does not have the authority or influence to act on them. Many valuable insights and ideas for improvement, which could lead to bottom line results, are left on the table.
- User input is limited to surveys and acceptance-testing. No time is taken to study user work and task flow. Users are not directly involved in the project. Significant areas of low user productivity remain unaddressed and not understood. When the "new" application is released, users roll their eyes over things that still aren't addressed, grimace, and move on.

b. IT Capabilities

- Programming Lead(s) and the Project Manager are very cautious about "promising" anything to business or users. The caution is born of the often encountered culture where the IT Department catches the blame for almost everything, and they are therefore very reticent to stick their necks out.

> We encountered this in a number of companies and it is no fault of the programming team. Painful experiences have taught them to be very cautious. Establishing a company culture that respects and understands the difficulties and challenges of the IT group goes a long way to alleviate this tendency. Until the IT team feels safe to embrace the challenge and promise of a well-designed and user-centered application, your company's results are going to fall short of their potential.

- Neither the Programming team nor the UI Developer is highly skilled in developing code for display and will often not attempt things that are well within the realm of the doable simply because their skills are not sufficiently developed.

> In our professional capacity as outsourced application developers, we often create display-coded templates which are quite complex in their behavior. Once we show the programming staff what can be done, they readily get behind more robust experience design.

c. User Needs

- No team members have direct contact with the users, let alone have dedicated User Representatives as a part of their team. User input is all indirect.

> You'll see this observation often (this lament, actually) throughout this report. Some teams simply do not have contact with users. Teams that embrace user research and user-testing for the first time have almost religious conversion experiences. We've seen team members learn more about what their application needs in one afternoon with users than they gained in months of team processes.

- Team members often say, *"We know what users want,"* when in fact, while they know much that the users want, they lack an understanding of 20% to 30% of the users' needs—which sometimes includes the most critical needs. Knowledge of user work and task flow is non-existent, or known partially by the Business Analyst(s) or Team Leader, but not incorporated into a methodical process.

CORE TEAM ROLES

Teams designed the most successful user strategy, user experience, and user interface where there was a core team composed of well-trained individuals who jointly and thoroughly executed the following functional roles:

Stakeholder Champion(s)
User Representative(s)
Programming Lead(s)
Business Analyst(s)
UI Developer(s)
Project Manager
Team Leader

We encountered a wide range of titles for all of the application development positions. We thought it would be most helpful to describe generic roles and give them generic titles. In the end what matters most is not job descriptions and titles, but that all the functions of these various roles are carried out appropriately (according to the scope, sequence, and phase of the application development process), and in concert with the other team members (following a methodical iterative process).

On some teams we studied, one person could (or needed to, due to a small team size) play more than one role. We saw some very effective teams that were as small as five people, including the programmers. The Team Leader and Project Manager roles are often combined on small projects. Team Leaders also frequently covered some, or all, of the Business Analyst role. Programmers sometimes also have the role of Business Analyst, and sometimes UI Developer. Sometimes the Team Leader is also the Stakeholder Champion. We have observed all these variations and more.

However, role combinations only worked well when there was a clear understanding that the individual was taking on distinct functional responsibilities and when the volume of work allowed the individual enough time to adequately perform both roles. When any one of these roles were missing, undervalued, or combined into one individual who could not adequately perform the multiple functions required by the role, user strategy, user experience, and user interface development suffered significantly.

On the other hand, we observed several teams that had more than one person in the same role. Frequently there was more than one Business Analyst, UI Developer and, especially, programmers. However, typically, when the application is too large for a team, rather than creating a larger team, the application is broken into tracks, products, or some other functional division, in which two or more teams are assembled that work separately.

When applications are developed by multiple teams, a higher level of organization is required. In this first version of our study, we were unable to address the challenges and best practices of managing multiple teams, though we would like to address this in our first revision. For example, there are various specialized teams which perform such tasks as usability-testing, user interface development, graphic design, and style guide development and maintenance, which serve multiple development teams. There is also, of course, higher level coordination required to manage several development teams, such as an overall product manager, application architects, database architects, etc. We would very much like to study this dynamic more and assess the best practices in use for our next version of this report.

However, in this report, we concentrated on the most effective team composition that most directly affects user strategy, user experience, and user interface development—those of the core team. If best practices are lacking in the core team's development process, no amount of specialized support or management will yield an excellent user strategy and user experience—the core team makes it or breaks it.

Readers of this report may be surprised to see no specific positions in our list of core team roles which are titled in relation to usability or experience design. There are three reasons for this. First, we chose the more established and generic title of UI Developer over a large number of variations which include the words usability or experience design. The title and the functions of the UI Developer position are changing as companies increasingly realize the importance of usability and experience design. We chose to go with the older, generic, more easily understood title of UI Developer because the new crop of titles has not quite resolved out into a commonly understood new title.

The second reason, and one we are very pleased to report, is because usability and experience design are increasingly seen less as functional positions as they are seen as skills that should be learned by most of the core team. In other words, the responsibility for usability and experience design is increasingly viewed as belonging to the whole team, not just to one position. The BA(s), UI Developer(s) and Team Leaders are getting more and more versed in the concepts and accompanying practices of user-centered design.

The third reason we chose to use the generic title UI Developer over newer variants of usability and experience design is that the newer crop of titles are often carried by in-company specialists whose role is to roam from project to project to provide help to all the core teams. We did not see where this was very effective. Either they become marginalized in their effectiveness because teams can take or leave their advice, or they become "interface police" and end up in an adversarial position in relation to the core teams. One Team Leader told us she *"Wouldn't touch the experience design team with a ten-foot-pole."*

In the end, we believe usability and experience design are skills composed of a number of specific practices which should be thoroughly present in as many core team members as possible.

1. STAKEHOLDER CHAMPION(S)

(Sometimes also referred to as Business Lead, Product Manager, Product Owner)

Stakeholder Champion(s) (SCs) bring focus and decision to the business aspects of application development. Often working as liaisons for a larger group, such as a steering group, or for management in general, Stakeholder Champions insure that bottom line business goals are optimized in the new application.

A pivotal position, the Stakeholder Champion can insure or destroy the chances that an application will be highly usable (user-centered) and highly productive (work- and task-flow oriented) simply by the fact that they have enormous say in the process. Typically, SCs can overrule anything of which they don't approve, or insist on anything they really want.

We chose to call them Stakeholder Champions because the best ones have a stake in the outcome of the application—it will help or hinder their career depending on how it comes out—and they can be champions to management to get funding, and evangelists to other influencers to win them to a user-centered approach.

The important thing we learned was that in teams without the Stakeholder Champion role represented, regardless of the title, or where the role was played poorly, applications suffered significantly.

Best Practices for the Stakeholder Champion(s)

a. Champions the users—wants them to have a tool that "really" works for them.

- Stakeholder Champions have enormous potential to influence a project for good or ill. The SC is usually a senior, if not the most senior, member of the team, frequently having more authority than the Team Leader. When the SC is passionate about satisfying the end user, resources and decisions flow to support that passion.
- They "believe in" usability and realize that if the needs of the user are shortchanged, then the ROI of the application will be shortchanged.
- Stakeholders tend to get involved with an application project only rarely. If they were involved all the time, they would cease being stakeholders, with their roots in the business processes of the company, and become application development team members, such as Business Analysts, instead. This means that stakeholders are rarely schooled in application development processes, and take their user-centered attitudes (or lack thereof) from management.

b. Views the process as an enormous opportunity to increase productivity.

- Superior application development is really composed of two parallel processes. One process is the creation of an application that supports user work and task flow. The other process is the improvement and streamlining of the user work and task flow itself. The best SCs are working as much with the operational opportunities to improve efficiency as they are with the development team process.

- The difference for the team between the stakeholders who "get" this, and those who don't, is enormous. In this age of information workers, the applications available to the workers by and large dictate task and workflow—thus efficiency and productivity. In a recent project we were able to convince the stakeholder to increase the screen "real estate" available to his staff by adding a second monitor. This was vital to the auditing process his staff was engaged in, which involved comparing documents to audits and allowed us to design screens that combined steps that were previously spread over steps. The users loved it. He was a hero both to his staff and management.

c. Champions the application to management—gets funding and resources.

- Every development process hits problems—delays, lack of resources, schedule issues, scope issues, and lack of funding. The best SCs win management approval for optimum solutions for all of the above. No other single person on the team has the reach and the clout to win management approval for so many aspects of the process vital to the creation of excellent user strategies and user experience.
- SCs can command the internal facilities and resources required to do ongoing user-testing, form focus groups and find end users outside the organization.
- SCs may report to or be a member of a "steering group" which has enormous influence over the goals and funding of the application. In one instance we observed, the steering committee was the Stakeholder Champion.

d. Deeply involved in the development process.

- The effective streamlining and improving of user work and task flow requires very granular involvement by the SC. Potential improvements in workflow need to be understood thoroughly in order for the SC to win managers, and upper-level managers, to real changes in operations. Without the SCs' championing of those changes, they won't happen. No one else on the team has the authority and influence necessary to effect operational changes to the extent an SC can.
- We often hear stakeholders comment in a group meeting, where task and workflow is being explored, "Why on earth are we doing it that way?" Stakeholders, and management in general, are once or twice removed from the day-to-day processes employed by their staffs. Often, when they really see workflow laid out on a white board they are able to appreciate the need for streamlining—and most importantly they have the authority to make it happen. Sometimes whole steps are eliminated from an application simply because the stakeholder knows it is no longer even necessary, or soon to be changed.

e. Doesn't hide when matters get technical.

- The best SCs are comfortable with the technology involved in the development of the application—from databases to display code. While they are seldom technologists themselves, they have no problem learning what they need to in order to participate in the process and make informed decisions.

  If the stakeholder has the most clout, as is often the case, then the SC is in the best

position to make sure that a user-centered application design approach doesn't get lost. However, if the SC turns away from crucial technical issues because they don't feel they can understand them, then they will not be sufficiently aware of the situation to guide it, because it is the stakeholder, and often only the stakeholder, who can insist on pushing the programming team in a direction that favors the user.

Normal Practices

a. The SC is several layers removed from the actual users and has little real awareness of their needs. Indeed they are not so much Stakeholder Champions as they are stakeholders, primarily concerned with budget and business requirements.

b. The SC is too busy or unaware of the opportunities a development process affords his or her business unit to streamline and improve work and task flow. The tendency is to want to make a few specific improvements that have repeatedly come up as "problems" (reported up the chain by mid-level managers) but not take on too much.

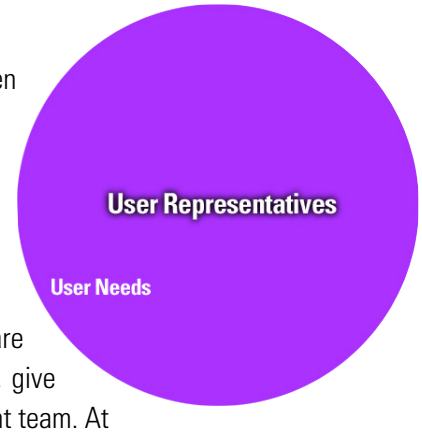
c. The SC is more interested in getting the application done on time and on budget than making it "a big deal."

d. The SC is unfamiliar with application development processes and allows himself or herself to be intimidated by the IT staff's objections and concerns as to what can or cannot be done. The SC is not comfortable with technology and therefore lets others talk him or her out of pushing for needed solutions because they would be difficult to do.

- As one stakeholder put it: *"I leave technology decisions to the technologists."*

e. The SC has massive amounts of business requirements documentation but no prioritization. The SC is not clear on what are the highest business priorities and relies on Business Analysts to "figure that out."

2. USER REPRESENTATIVES



User representatives should be actual end users. It surprised us how often this was not the case. Some teams have referred to user input or user feedback they have received, which is actually from the supervisors or the managers of end users, or is indirect feedback such as help desk logs. There is nothing more revealing than, and no substitute for, talking with actual users, and when possible, observing them in their own work space.

It is also very effective to develop a consistent group of end users that are observed and consulted often, participate in development process steps, give iterative feedback and who become, in many ways, part of the development team. At specific points in the development process, new users should also be observed and consulted who are not biased by previous exposure to the development process. However, a consistent group of users who are consulted on work and task flow especially, is invaluable. As a group they are unmatched content area experts and have the most valuable feedback on screen design of anyone on the team.

User Representatives are the most important group to satisfy—whether in an internal, B2B, or a B2C application—they are the customers, the ones for whom the whole development process is performed. However, users are the most powerless. The rest of the team has all the authority, the budget, and the ear of management. If the team itself doesn't put the users first, by managing and truly deferring on important points to an effective group of User Representatives, then business goals and IT concerns will dominate the development process—the single most common failing of all development teams.

Best Practices for User Representatives

a. Maintain a consistent and experienced User Representative group for planning and initial development.

- Power users are preferred as the core of the ongoing User Representatives group. Less experienced users should be included in new groups of users for prototype-testing or usability-testing to insure that the perspective of a novice user is considered, but the most effective group is made up of experienced users of an existing application being improved, senior employees in the business process the application will be supporting, and experienced users of computers in general.
- An effective group of User Representatives begin to feel like part of the core development team and in the best development processes they are treated as such. They have a seat at the table.
- Some User Representatives will, as is natural in any group, be more effective for you than others. We have seen some User Reps get so involved that they began sending their own wireframes—good ones, too—to the development team. Others are more quiet, but prove their worth with specific knowledge of the tasks they do day in and day out.
- An effective group of User Representatives who are involved with the development process become grassroots evangelists for the new application or new release and significantly increase buy in when the application is finally released to end users.

- Managers or supervisors who are occasional users of the existing application being improved, or managers of the business process which the new application will be supporting, are good for general user input but not for input on fine points of use. They have invaluable input from a business point of view but not as end users.
- In some instances we encountered too much concern from managers about what the end users might say and therefore resisted having them involved in the process. Perhaps they are concerned it would reflect badly on them as managers, perhaps they want to make sure nothing gets developed that they don't know about. In any event, we observed that careful explanation of the purpose of the User Representatives allayed their concerns.

b. User representatives, to be most effective, must be significantly involved in the process.

- It is nearly impossible to get highly valuable input on a complex application, whether in the initial planning stages or in later prototype-testing, if users are drawn on on an hour-here-and-an-hour-there basis. The most effective User Representative groups were involved in the development process from 80 to 200 hours over the course of the development process. Because of their long-term involvement, their input becomes significant to core user strategy and user experience development.
- The user group representatives participate in work- and task-flow workshops, visual screen design sessions, and are an integral part of the iterative feedback process as use cases and visual representations of the screens, such as wireframes, are refined.
- It is important to get informed approval from management to allow these User Representatives to take this much time away from their regular work. Once the project is launched you don't want to have your User Representatives withdrawn, or others substituted for them. Continuity of input is very important.
- By and large their main contribution is in their reactions to ideas, wireframes, etc., which the core team have developed. However, enthusiastic User Representatives sometimes contribute significant ideas for user strategy, user experience, and user interface improvements.

c. New User Representatives, asked to test prototypes, also need significant exposure time to the prototype to give meaningful feedback.

- Usability-testing, confined to a few screens, or a few tasks, can be accomplished with short time commitments from new users, but if the need is for considered feedback on a complex application designed to support a business process, users need significant time, and training, to give highly valuable feedback.

Normal Practices

a. Unfortunately, the most normal practice is for there to be no direct user representation on the core team at all. User input is gathered through surveys, indirect input (such as help desk logs), and through the impressions of supervisors or managers (who are at least one step removed from direct user experience and whose bias is toward business goals rather than user experience).

- We could, alas, provide you story after story illustrating this point. Suffice it to say this is the

single most important factor that insures or destroys true user-centered design. This is the great divide—companies which embrace direct user research and user participation in the development process are well on their way to creating great applications—companies which do not embrace their users' knowledge are headed for poor results.

- When users are directly observed and consulted, their feedback is still channeled to the core team through another team member such as a Business Analyst, UI Developer, or Team Leader. In the realm of B2C applications this is necessary because the team cannot arrange lengthy end-user involvement, which can be arranged when developing for internal or B2B users. However, it is common practice for internal development projects, as well, not to directly involve User Representatives on an ongoing basis even when it is feasible.

c. Most teams, due largely to inexperience (or the wrong experience), and to having no effective methodology, have the attitude that users are a problem. They will say that users, *"Don't really know what they want," "Can only point out simple problems"* or they *"Actually get in the way"* of good solutions.

- One comment we heard was: *"I have no contact with actual users and count myself fortunate for that."*

> We have observed, to some extent among our study development teams, but significantly in our professional practice, that teams are often frustrated by user input. If they receive negative input regarding an application that they themselves developed, the team members will often say, *"They just don't get it. Everything they want is in the application. Why can't they just learn to use it."* Users can, on an emotional level, become the enemy. It is easy to appreciate how odd this attitude is from the comfortable distance of analysis, but when teams have given long hours and great effort to an application and it is roundly criticized, it is hard to stay positive about those "idiot" users. The remedy is for team members to develop an application where systematic, methodical, and significant user input was gathered and incorporated into the development process—and to experience the hero status developing a killer application brings to the team.

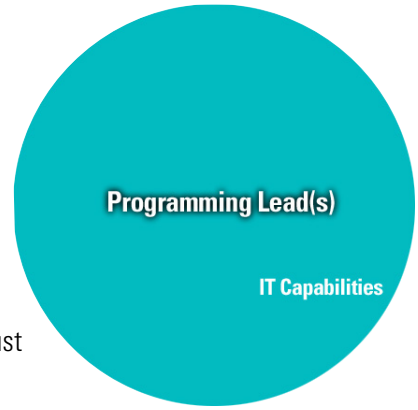
d. Many development teams are weighted toward business or IT. If weighted toward business there is often a blurring of the distinction between business goals and user needs, to the point where no distinction between them is made at all. If weighted toward IT, there is a strong tendency to think that the final user interface design phase comes at the end of the development process, therefore direct user feedback is only needed for prototype-testing or acceptance-testing when, unfortunately, user feedback can have little or no impact on the application.

- This is often where companies can really benefit from bringing in a user-centered application design team from outside the company. The outsourced team organizes and champions the users, keeps their perspective uppermost and provides training and experience for the internal teams to see how user input and User Representatives can be brought into the process.

e. Users are brought in for an hour or so of feedback and testing, or for participation in focus groups, in order to give feedback on a major application in development. By the end of their session they can typically only form superficial opinions on the application. They do not have enough time to perceive deeper user strategy or user experience features of the application and are limited to comments on the user interface only.

- A lot of user-testing or the user input process can become a waste of time. We've seen a lot of user tests, acceptance tests, prototype tests, pilots, etc., where users are asked to give feedback on what to them appears to be an avalanche of new application information. They barely have time to come up to speed on the most obvious elements of the application, such as how the navigation system works, yet are asked to give broad-ranging feedback on user experience and user strategy covering five screens. You might as well not bother. Better to go into the application in depth, over days if necessary, with one or two users than get a little superficial input from large groups.

3. BEST PRACTICES FOR PROGRAMMING LEAD(S)



(Sometimes referred to as Application Architect, Application Designer)

The Programming Lead(s) acts as the bridge between the front-end process and back-end implementation. The best Programming Lead(s) facilitate a full flow of communication in a rapidly iterative process. In the early phases of development, the core team needs to know, as definitively as possible, whether key user strategy and user experience directions are "doable." (Considerations that determine whether something is doable include not just technology issues, but budget, staffing, and schedule constraints as well.)

There is nothing more damaging to the user strategy and user experience planning effort than to find out too far into the process that key features and functions cannot be implemented. The end result of the changes required to cope with this late-breaking information is nearly always a blurry, confusing, and ineffective approximation of the original user strategy and user experience, resulting from a number of hasty compromise decisions, or independent decisions made by coders who are under tight time constraints. Had the team known in the planning phases that certain features or functions could not be implemented, alternative user strategies and user experience could have been developed, which were perhaps not as dynamic or feature-rich, but which would be clear, useable, and as productive as possible.

In large projects there is often more than one Programming Lead involved. The best results come when the single team lead, or multiple leads, can answer all the questions regarding data, coding, and infrastructure. If back-end development is outsourced, the best teams manage to get significant onsite involvement from at least one Programming Lead from the outsourcing company during the planning phases of the project.

When these best practices are in place, handoffs from the planning phases to the implementation phases go smoothly, fewer iterations are required, QA identifies fewer problems, and the entire implementation process goes more quickly.

Best Practices for Programming Lead(s)

a. Involved throughout the user strategy and user experience development process (especially if outsourced).

- While it may appear to be wasteful to allocate valuable lead programmer resources to participate in the planning phases of the application development process, the net result in end phase implementation efficiency more than make up for the initial resource allocation.
  - > We were told of one outsourced programming firm endlessly revving a section of an application (over 50 times) burning up valuable time for both the outsourcing team and the in-house team. When we explored the problem we discovered that the programming team was working only from documentation. They had no real and immediate sense of what the application was for, or how users would actually use it, or any understanding of the key drivers of user strategy and user experience. Processes were changed, a senior member of the outsourcing team became an

onsite resource, and the next section went through in half a dozen revs. From what we have observed and studied, this is the most significant solution to typical problems encountered with outsourcing and off-shoring. Early, onsite involvement from key players in the outsourced team insures the optimum handoff to the coding and implementation phase of the development cycle.

b. A key decision-maker: such as system designer, system architect, or lead.

- The best teams are allocated a significant resource, not just a go-between, who reports back to the real decision-maker, but the person who can truly deliver the code and data to support the application being planned.
  - > System designers, system architects (or any of the other titles of lead strategists for programming) who are more interested in an efficient code base than in knowing and understanding the business process goals and user needs of the application they are developing, do not serve their companies well. Bottom line return on investment comes from highly usable and highly productive application user strategies, user experience, and user interfaces supported by solid code—not the other way around.
    - - We interviewed a team member who was responsible for the display code development and was also a programmer. We inquired about user interface design and how his team approached it, only to have him go on and on, with great enthusiasm, about their new development environment with its accompanying interface toolkit. For him, user interface design was about the tool box of module styles, navigation controls, icons, colors, and shapes—and had nothing to do with how users want to experience the application.

c. Comfortable with a "top down," iterative approach.

- The best Programming Lead(s) fully engage in the planning process which requires multiple cycles of conceive-idea to check-feasibility-of-idea to refinement-of-idea, and they do so vigorously and rapidly.
  - > In order to contribute meaningfully to the iterations, the Programming Lead(s) needs to be comfortable with not knowing everything definitively, to work from a top-level perspective, and have enough experience to make reliable decisions about whether something is doable before all the details are known.

d. Involved in creation of design specification.

- The best handoffs to the back-end coding and implementation team members from the front-end planning team members are accomplished when both back-end and front-end team members co-create the design specification.

Normal Practices

a. Programming input comes solely through interactions between the Team Leader and someone in the programming group. The Team Leader does the best he or she can, supported by whatever documentation he or she can put together, to describe user strategies and user experience ideas for the purpose of getting technical approval.

- As a result very few iterations take place.

b. The Programming Lead who is the real decision maker doesn't put his or her mind to the application under development until the front-end planning process is completed.

- The Programming Lead is fully allocated to other projects.

> We sometimes observed an attitude that not only were the Programming Leads not available by schedule, but that they really didn't take the need seriously. Their attitude seemed to be that whatever the front-end planning process came up with was fine—because the final decisions were going to be made by the programming team anyway. This always resulted in the front-end planning suffering considerably during programming and implementation. The application planning undergoes a "death by a thousand cuts"—an accumulation of small decisions by the programming team that depart from the original user experience and user interface design.

c. Programming leads are too cautious (either by nature or due to company culture) to commit to what can and cannot be implemented until they get the completed application user strategy, user experience, user interface, and database schema. The most common thing you hear said is, "We'll see at the end what we can and can't include in this build."

- There are very good reasons why programmers feel this caution. User strategy may appear to be relatively simple, where supporting the strategy can be very complex. There are many factors that need to be weighed—available resources, schedule, performance, security, reliability, ongoing support—just to name a few. This is why iterative development is the most effective.
- However, there is one very bad reason why programmers feel this caution. They are often schooled in a bottom-up process. Typically, programming teams take every aspect of what the application needs to be able to deliver and breaks it down into its smallest parts (data primarily), and then re-assembled upward, so to speak, into a functional whole. This may insure that it can be implemented efficiently, and that no piece of data is forgotten, but it often eliminates any effectiveness of user strategy and user experience planning. Typically, in the bottom-up process, the user interface, and therefore user experience and user strategy as well, are the last things developed, which nearly always results in a data-centric application as opposed to a user-centric application.

> The loser of the tug-of-war between the bottom-up approach which insures thorough attention to detail, and the top-down approach which insures thorough attention to user strategy and user experience, is the user. A completely top-down approach is the only way we observed to develop a user-centered application that can be highly usable and highly productive.

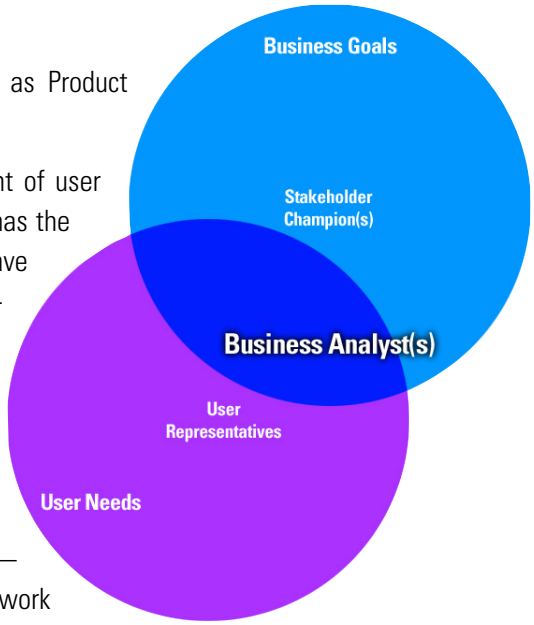
d. The Programming Lead(s) waits for a design specification document, which signals the beginning of the back-end coding and implementation process, and have not seen or commented on it in any previous version.

e. The programming culture is systems— and code-oriented. Programmers often pride themselves on meeting the abstract challenge of writing "clean, well-behaved code" that is easy to extend and maintain over time.

- However, user needs are sometimes messy. Effective user-experience design often pushes the envelope, creating code complexities, which often become difficult to maintain over time. However, user experience conventions which could result in real operational efficiencies on the business side (thus saving or making real and significant money) often get lost in a short-sighted decision for code efficiency, which, in the end, will not do as much for the company's bottom line as the users' gain in productivity would. Programming groups which are too code-oriented will often make the case that something is too difficult when in fact it simply doesn't fit neatly into how they want to structure the code.

We have seen many a promising user experience idea die in this way. When the programming group is highly involved with end users, however, this rarely happens, because they have a real and immediate sense of the high value a particular user experience might have for the user, and go out of their way to figure out a way to implement it.

4. BUSINESS ANALYST(S)



(Sometimes referred to as Business Process Analysts, rarely as Product Managers, Product Owners)

The Business Analyst(s) (BA) role is crucial to the development of user strategy and user experience development. However, this role has the widest variation in responsibilities and skill sets of any we have studied. We consistently observed that when there is a significant lack in quality (or even an absence altogether) of user strategy and user experience development, it is usually because the Business Analyst(s) or the Team Leader did not perform, or did not perform well, critical steps in the development process.

The best BA(s) become subject matter experts in two realms—business goals and business process, and user needs and user work and task flow. The Business Analyst(s) then synthesizes this knowledge, along with input and discussion in core team strategy sessions, to develop top level user strategies, and subsequently, user experience. The best Business Analysts are able to combine global thinking with deep knowledge of detail and express that through visual design of screens, such as wireframes—representations of actual user screens—and textual documentation, such as use cases, and features documents.

The best BA(s) works very closely with the Team Leader during the user strategy and user experience development phase of the project. The BA(s) may often lead work- and task-flow workshops, as well as the wireframe development workshops, though that role is also often done by the Team Leader. Good synergy between the BA(s) and the Team Leader results in innovative and excellent user strategies and user experience.

The best BA(s) makes sure that user strategy and user experience decisions and conventions remain consistent all the way through the development process. The BA(s) is a "user champion" and sometimes lead, but are definitely involved in user research, paper prototype-testing, focus groups, and prototype- and acceptance-testing.

The best BA(s) is involved in, and may lead, the business requirements-gathering process and work closely with the Stakeholder Champion and Team Leader in this process.

The BA(s) is often responsible for developing and maintaining core documentation, including the business requirements doc, features doc, wireframes, and use cases, and participate significantly in the development of the design specification (See Documentation).

Best Practices for Business Analyst(s)

a. Knows the users' needs, and work and task flow, in detail.

- The best BA(s) conducts thorough user research and work- and task-flow workshops in order to dive deeply into the world of the intended users of the application under development.
  - > Gaining a detailed understanding of the users' work and task flow is the core requirement in order for truly user-centered applications to be designed. Application

users need to accomplish tasks that fit into a larger workflow. Users often perform these tasks repeatedly. If the application doesn't "work the way they work" it will always be a drag on productivity and an ongoing point of frustration for the user.

- We interviewed several BAs who had risen to their current position through the training department, where they gained commitment to the importance of the actual user perspective. They saw firsthand that no matter what the development team says an application can do, unless the users can do it easily and efficiently, they will not give an application a good rating.

b. Champions the users' perspectives and needs.

- Even with User Representatives functioning as part of the development team, the users are not always present in the process and rarely have the "clout" to push for their point of view. The BA(s) understands that if he or she doesn't constantly push for, and vigilantly defend, the users' perspectives, the business and IT influences will naturally predominate, simply because of the fact that they are well represented. (On the best teams the UI Developer and the Team Leader are also fervent user champions.)

c. Knows the business goals and priorities, and the desired business process improvements, in detail.

- Working closely with the Stakeholder Champion, the BA(s) can often be instrumental in suggesting operational improvements in the business process being supported by the new application.

> The best BA(s) we observed and interviewed were equally at home discussing business goals and business processes as development practices. Good BAs are worth their weight in gold to their company because they are in the pivotal position of transferring operational efficiencies into the new application.

d. Thinks strategically and yet is grounded in detail.

- The BA(s) is in point position to develop user strategy and user experience conventions. The best combine art and science. The art is in visual screen design, such as wireframes. Creating innovative diagrammatic representations of user interfaces (wireframes) requires a creative mind. But to be successful and ultimately useable, the wireframes have to be grounded in the science surrounding the process—display coding, detailed business goals, and detailed user needs.

e. Has significant influence and authority within the team.

- Equal to, and sometimes greater than the Team Leader, the BA(s) often has the most influence on the development of user strategies and user experience and thus significantly affects bottom line results through usability, productivity, and streamlining operational efficiencies through their work with the Stakeholder Champion. The BA(s) can be a power player.

> This of course varies from team to team. There are usually one or two strong leaders on the team who end up guiding and informing the creation of the user strategy and user experience design. The Team Leader is often a current or former BA.

f. Has highly developed skills for creating visual designs of screens such as wireframes.

- The BA(s) often brings the screens to life. (UI Developers and Team Leaders often play this role, too, instead of, or as well as, a BA.) In the end what is being developed will be experienced as a visual interface by the end users. Users don't see the code. Users don't see the database schema. Users see screens. The best BA(s) communicates through visual designs of screens primarily and text secondarily. They are often comfortable with Photoshop, and if the UI is already fully developed, can create screen designs with pixel-level accuracy in Photoshop, using the existing UI guidelines.
  - > The best BA(s) draws in a small core of team members to a rapidly iterative, very creative process for visual screen design. Other BAs, the Team Leader, and the UI Developer(s) are often the creative crucible from which emerges the killer ideas for user experience.
    - A small group of appropriately skilled people, with a whiteboard and a relaxed period of time, is often the single most important element that insures excellent user strategy and user experience. It doesn't really matter whether it is the BA or the Team Leader or another strong player such as a UI Developer who is the catalyst for these sessions as long as the work takes place.

g. Creates fully realized use cases.

- The best use cases (best because they communicate the most fully) are combinations of wire frames, flow diagrams, tables, and textual annotation. The BA(s) may function as the use case assembler, pulling in documentation from other members of the team. The best BA(s) create minimally-sized documents that convey the maximum amount of information.
  - > We observed wide variations on how use cases were written and how they were used. For “clean slate” application development projects they are almost always considered necessary. For major releases of existing applications, teams often forego use case development except in certain areas. For incremental development between major releases use cases were almost never used except as reference points from previous development. The most effective use cases we studied were primarily composed of wireframes, annotated by text, flow diagrams, and data tables.

h. Knows the science of usability.

- The BA(s) may not be best used for the highly detailed usability-testing of existing application screens. Other team members, or specialized teams, may be better suited to conduct usability analysis that leads to incremental improvements in the application. However, thorough grounding in the principles of usability and the methodology of usability-testing adds to the BA(s) ability to create highly usable applications.

i. Understands programming well enough to defend user strategy persuasively.

- The BA(s) often needs to fight to the end for really effective user experience ideas. The BA(s) is best able to do this if they are aware of the programming side of the process—especially display coding.

Normal Practices

a. The BA(s) is primarily a collector and documenter of business process information and has no strategic development role (or abilities).

- Unfortunately we saw this all too often. The BA(s) was a walking excel spreadsheet and could speak to what the business goals were in meetings, but they played no strategic role in forming the strategies of the application, and rarely had any skills at creating wireframes. In teams that performed poorly, it was often the BA role that was most lacking.

b. No strong BA(s) means the user strategy and user experience development role is missing and is "covered" by the Team Leader or sometimes by the UI Developer.

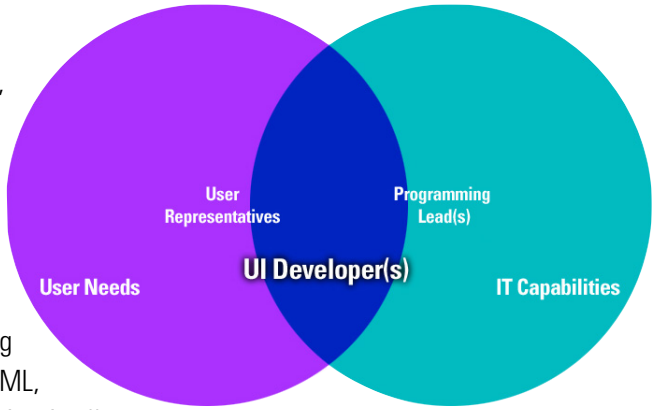
- This often results in no formal process of user strategy or user experience development and it gets done ad hoc, at various times, and in various phases of the development process. The end result is neither cohesive nor consistent.

c. The visual design of screens is often performed as an almost "back of the napkin" or casual whiteboard exercise rather than a core process of development and documentation because there is no one on the team who knows how to do them well. Almost all the development project's documentation is a textual description of a visual experience.

- The end result is usually that the actual user experience design, and the user interface design process is done by the programmers at the very end of the development process, too late for significant iterative improvement and a cohesive user strategy is barely evident.
  - > Programmers are not schooled in experience design or user interface development, even though they will often tell you they "can do it." The poorest performing applications, judged by the criteria of user satisfaction, were the ones where the team left the user experience and user interface design to the programming team. Not only are programmers typically not schooled in experience design and interface design, through the process of programming the application they become way too close to it to have any perspective similar to the average user.

d. The BA(s) often comes from a programming background and will sometimes fall into a habit of "pre-programming" as they do business requirements-gathering, visual screen design, or use case development. Streamlining code takes precedence over streamlining user workflow.

5. UI DEVELOPER(S)



(Sometimes referred to as Experience Designers, Information Architects, Usability Specialists, Interface Designers)

The UI Developer(s)'s most important function is to bring state-of-the-art knowledge to developing the display code. Supporting display code for multiple browser versions, sometimes for multiple operating systems, using multiple display code types ((X)HTML, Javascript, DHTML, XML, Java, Flash), and integrating it all into a particular development environment (such as .net) requires a well-developed skill set to be successful. The complexity and challenge of this skill set is often underestimated, which results in significant problems down the line, which in turn result in very poor tradeoffs in functionality when things don't work. Or, teams undervalue the contribution an excellent UI Developer(s) can bring to the application. Many features that users really love come about because the UI Developer(s)—as opposed to the programmers—can pull off the innovative solutions envisioned by the rest of the team.

The UI Developer(s) tends to have either a graphic design orientation or a coding orientation. If the former, it is essential that the UI Developer(s) has mastered the intricacies of display code as well as have good design sensibilities. If the latter, it is important that the UI Developer(s) has a sense of design or that there is more than one UI Developer on the team with balancing graphic design skills.

A high-level graphic designer can be brought in to successfully create an excellent "look and feel," typography, layout, and icon development for an application which complements the coding skills of the UI Developer(s). However, a high-level display code expert cannot successfully be brought in to augment the coding skills of less experienced UI Developer(s).

The best UI Developer(s) often has training in usability-testing and analysis, and will often be a leader of (or be significantly involved in) user research.

The UI Developer(s) is also the champion and defender of UI standards which are required for the application under development, such as corporate guidelines, and any previously decided upon standards for icons, use of color, typography, user experience conventions, etc. The UI Developer(s) can extend a UI look and feel to new applications (though high-level graphic designers are often needed for major revisions, icon development, and new user experience styles).

The UI Developer(s) may also create and maintain (or help create and maintain) a user interface/user experience style guide which includes not only graphic guidelines, user experience styles and conventions, but up-to-date display code as well.

Best Practices for UI Developers

a. Has high-level display code skills.

- The best UI Developer(s) can render a Photoshop file of a screen design into working display code with pixel-perfect accuracy across a wide range of browsers and operating systems. A highly usable screen design is successful because it expresses the user strategy and user experience down to every last detail.
  - > Additionally, the best UI Developer(s) is highly skilled at coding client-side interactive elements using Javascript, DHTML, and (though rarely) Flash. Some of the most productive interactive features in web-based applications are client-side, which reduce the number of server calls and therefore speed up the users' experience.
    - - In the appropriate situation, Javascript or DHTML can be used to resize elements (adding or eliminating data) or provide small wizard-like experiences (such as a sequential menu choice—each choice determines the next available choices), without the need to make a server call, thus increasing the speed of the users' experience.

- The best UI Developer(s) provides crucial input during visual screen designing to insure that the wireframe designs being considered can, in fact, be coded.
  - > The best UI Developer(s) will often quickly "dummy up" display code to prove the viability of innovative ideas while those ideas are still being reviewed and considered.
    - - This affects two dynamic processes taking place simultaneously. First it affects a quickly moving process of screen design. The experience design process, once it reaches the wireframing stage, considers a lot of creative solutions that are heavily dependent on display code and back-end code to work. The experience design crew is much more effective if they can move forward with quick assessments of the "doability" of their ideas. The second process this affects is more subtle, but no less important, and that is the programming teams' acceptance of the screen designs. A good UI Developer can engender a lot of confidence among the programmers that the experience designers' "wild ideas" can be done.

b. Has solid graphic design skills—can extend existing UI or faithfully follow company-wide UI guidelines.

- The best UI Developer(s) has some graphic design experience—whether they actually worked as graphic designers or picked up graphic design skills along the way.
- The best UI Developer(s) is very comfortable using Photoshop, and is able to match an established style, when creating graphic elements such as banners, borders, headers, buttons and, sometimes, icons.
  - > Icon development, as well as the original establishment of the overall application look and feel, is often best left to a high-level designer.

- The best UI Developer(s) is thoroughly aware of existing UI Guidelines and is able to keep the application development process within the parameters of existing guidelines, or know when the existing guidelines can be extended.

c. Has cross-discipline experience in user experience design—can develop visual designs of screens, such as wireframes.

- Visual design of screens for user experience design is often the primary responsibility of the Business Analyst(s), or Team leader; however, it can be extremely beneficial to the development process if the UI Developer(s) is capable of participating in a wireframe-style visual design of screens.
  - > Business Analyst(s) very much need help to manage the volume of designs which need to be developed during the heat and the heart of the visual screen design process for a new application especially, but even for a major new release.
  - > There is also a very valuable synergy that can take place between the Business Analyst(s), Team Leader, and the UI Developer during the visual screen design process, which can significantly improve the user experience and doubly insure the "code-ability" of the designs.

d. Understands programming well enough to persuasively defend user strategy, user experience, and the proper integration of display code.

- Many of the best UI Developer(s) comes out of a programming background, or has picked up a lot of programming knowledge through their careers. Display code, while a distinct skill set from other programming tasks, nonetheless must integrate seamlessly with back-end programming. The best UI Developer(s) is able to be a "code bridge" between the front-end process and the back-end programming process, by proving out display-code solutions that are of concern to the programming group—to get buy in and blessings from the Programming Lead(s).

Normal Practices

a. The UI Developer(s) has only passable display code development skills.

- The programmers actually do the main display coding. And in turn, the programmer's display code skills are adequate but not extensive, and they tend to choose very "safe" display code solutions, which work against innovative user experience.
- The UI Developer(s) develops display code primarily for small, incremental changes in an existing application, freeing programmers from the same task.

b. Display coding is not considered a special skill set. The programmers all do their own display code in the flow of their main code work. It is not considered to be that important. The real work is the back-end code. In these situations the UI Developer(s) (if the role exists at all) plays more of a graphic desig role to provide graphic elements to the programmers as they need them.

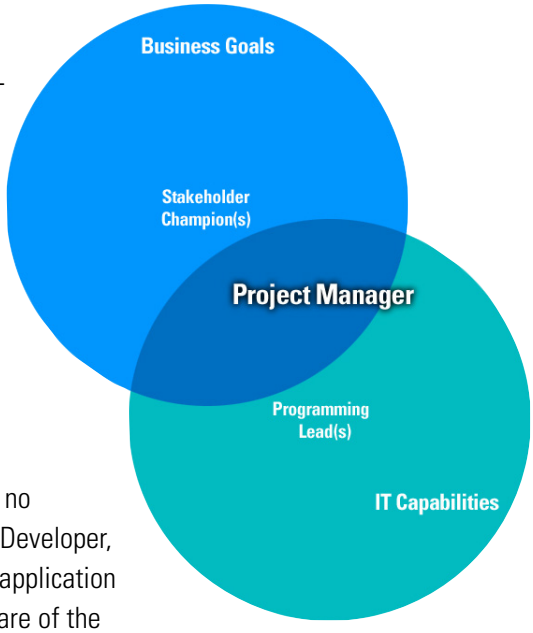
c. User interface and user experience style guides exist, but are often ignored (even forgotten). Frequently the UI Developer(s) is an entry-level role, and as a result, the UI Developer(s) does not have the authority to enforce or influence the use of the existing styles.

- The subtleties of good graphic design, typography, and module development are often not appreciated as having a significant affect on user satisfaction, usability, and productivity. Many applications look "clunky," but development teams and the managers of those teams simply think users will get used to them and they say things like, "We're not Apple, after all." The reality is that a significant portion of what makes applications usable is accomplished by shapes, colors, and typography. Internal application users typically spend hours a day using these applications. They are constantly scanning pages for information and making choices of where to click next. Good graphic design can make a night-and-day difference to the ease with which these everyday users move through the applications. One user told us that by the middle of the afternoon he wanted to "throw the computer out the window" and was seriously concerned about his eyesight. These are very real issues which companies would do well to pay more attention to.

d. The UI Developer(s)'s role, if played by a member of the programming team who is more versed in the traditional bottom-up development process, sees the user interface ("the UI") as "building blocks" that he or she is responsible for updating and making available to the programming team.

- UI toolkits and style guides are good news and bad news where good user experience and user interface design are concerned. The good news is that an established set of UI building blocks, assuming for the moment that they are good building blocks, insures a consistency of look and feel to an application which is reassuring to users. The bad news is that teams often mistakenly think that having the building blocks is all they need to create highly usable applications. They are lulled into a sense of confidence by the fact that everything appears to look good. The reality is that the building blocks are just like Lego blocks. It is possible to make a really cool toy out of them, but equally possible to make a misshapen mess.

6. PROJECT MANAGER



For the purposes of this study, we have concentrated on the best practices of the Project Manager that lead to superior user strategy, user experience, and user interface development, rather than trying to address all the facets of what leads to excellence in this role.

The Project Manager role is familiar to everyone who works in application development and because of that, we will not attempt an overall description of the role.

Best Practices for Project Managers

a. Pro-active about the project.

- The best Project Managers, even if they play no particular skill role (e.g. Business Analyst, UI Developer, Programmer) and do not work directly on the application under development, are nonetheless fully aware of the business goals and user needs that inform the development process, and participate in the user strategy, user experience, and user interface development process.
  - > The Project Manager is the only person on the team, other than the Team Leader, who will interact directly with all team members, as such, the Project Manager is in a unique position to communicate and facilitate communication among team members that leads to a "holistic" understanding of the application under development.
  - > An informed and enthusiastic Project Manager acts as an extra set of eyes and ears to help the Team Leader keep the process flowing in the right direction and to head off problems, or potential time-wasting activities that are off point or ineffective, before they happen.
  - > The Project Manager also frees the Team Leader from time, resource, and communication coordination activities which allows the Team Leader to focus on creative solutions—but only if they are fully aware of the project needs and project flow.

b. Integral to "selling" the user strategy and user experience to the programming team during back-end development.

- The Project Manager role spans the timeline of the application development project from beginning to end. As such, in the programming, testing, and deployment phases of the project, the Project Manager may be the only person who participated in the front-end planning process who is regularly available to the back-end team. The best Project Managers are able to keep the user strategy, user experience, and user interface whole and compromised as little as possible by the inevitable changes that are required during programming, by constantly selling and "evangelizing" the front-end solution.
  - > A good Project Manager is often the best insurance against the "death of a thousand cuts" which a good plan can sometimes suffer as it is coded and implemented. If, as is often the case, the Team Leader, UI Developer(s), BA(s) and

stakeholder have been allocated to other projects, the Project Manager is the only one dealing day-to-day with the programming team, whether in-house or outsourced, during the mid-to-last phases of the project. Programming team members are known to simply change things in the interface as they go along. We have often seen a fully implemented screen where all of the text is bold and two or three points larger in size. When asked why the change was made, the programmer replied that they "liked it better." More often, however, many of the thousand cuts are changes made because the programmer was unable to make the screen look like the wireframe. In some cases, this needs to be dealt with by bringing other team members back in for short bursts to make a new experience design decision, and sometimes the programmer simply doesn't know how to render the original design and the Project Manager needs to gently insist on conformance with the wireframe. In any case, if the Project Manager knows and embraces the application design, he or she can head off lots of these "cuts," or minimize their impact on the application.

c. Project Manager for entire development team—not just Project Manager for programming team.

- We often encounter a sense of "ownership" in application development projects. Where "ownership" is felt to be ITs, there is a sense that the Project Manager is being loaned out to the development team for the duration of the front-end planning process, but his or her loyalties lie with the IT group. The best projects have a sense of joint ownership between IT and Business, and the Project Manager is neutral.
- Even if, as is most often the case, the Project Manager is based in the IT department, the bes Project Managers are comfortable in all phases of the project—from front-end planning, through coding, implementation, and deployment.

Normal Practices

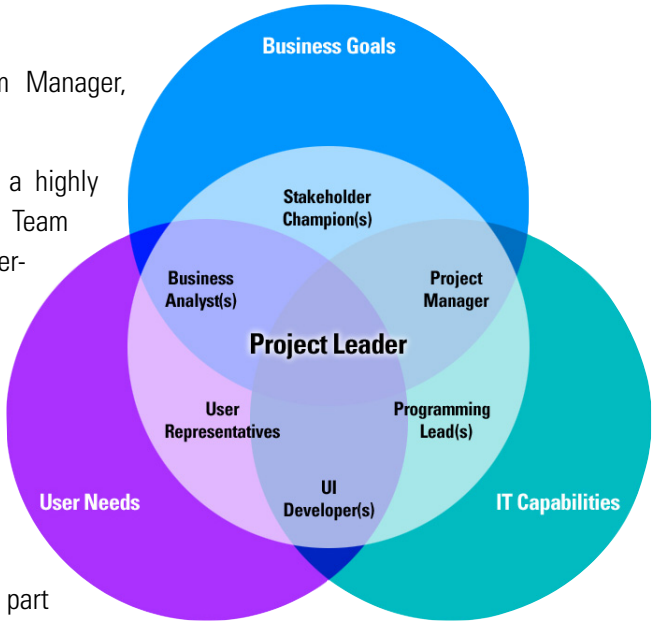
a. Project Managers, when not playing the dual role of Project Manager and Team Leader, are often uninvolved in the actual process of developing user strategy, user experience, and user interface.

- The Project Manager role requires a certain neutrality, but this often becomes indifference. The Project Manager doesn't "buy in" to the user strategies, paradigms, and conventions that are developed during the front-end process (due to his or her lack of involvement) and therefore is not committed to defending them during the crucial back-end implementation phase of the project when he or she is in the most effective position to do so.

b. Project Managers, whose primary role is to stay on schedule and on budget, do not want to add complexity to an already complex process, and so avoid pushing for the more challenging solutions that have come out of the front-end planning process.

- Features and user experience conventions are often not implemented because the programming team finds them "too difficult," or they "don't want to maintain the code." There are often very good reasons for this. However, the Project Manager who is uninvolved in or indifferent to the business and user goals for the application, doesn't know which features should be pushed forward, and which, having a lesser priority from a business and user point of view, could be dropped. Therefore the decision is usually made on the basis of programming efficiency.

7. TEAM LEADER



(Sometimes referred to as Project Lead, Program Manager, Product Manager, Product Owner, Project Manager)

No other position is more important to designing a highly usable and highly productive application than the Team Leader. The best Team Leaders are committed to user-centered design. Period. If the Team Leader isn't committed to designing a great application for the user, the rest of the team cannot make up for that lack. The Team Leader is the one person best positioned to make the user strategy, user experience, and user interface, in the words of Steve Jobs, "insanely great" for the end user.

Part evangelist, part cheerleader, part visionary, part hands-on creative designer, part business-savvy analyst, and part geek, the Team Leader is often a very experienced jack-of-all-trades. Their view of the application is the broadest and the deepest.

The best Team Leader drives the development of user strategy and user experience, going deeply into user needs, user work and task flow, and business requirements and makes sure all factors are creatively and innovatively supported in the application.

More than any other team role, the hands-on functions of the Team Leader are less important than their overall impact on the application development process. Team Leaders come from many different backgrounds: we've seen Team Leaders from training, sales, operations, IT, and programming, just to name a few.

As strange as it sounds, sometimes the Team Leaders (as far as developing user strategy, user experience, and user interface are concerned) are not "officially" considered to be the Team Leader. Sometimes the "official" Team Leader is floating too high above the process and only dips into the application development process from time to time—but the real "Team Leader," in the sense that we mean, is perhaps a Business Analyst or a UI Developer or, in some cases, the Project Manager. The title doesn't matter—but on every project you will find that there is one person who is the real driver of the user strategy, user experience, and user interface design process—whatever they are called, this section is for them.

Best Practices for the Team Leader

a. Brings the development process to an intense focus on the user.

- The best Team Leaders are passionately committed to user-centered design methodology, and are tireless champions of the end user. (This sounds obvious, but we rarely encountered it.)
- The best Team Leaders spend significant time with users doing user research and conducting work- and task-flow workshops. (See Gathering User Requirements)

b. Champions the best mix of business, IT, and user opportunities.

- Every application development project is a blend of three factors: business goals, IT capabilities, and user needs. Each factor presents opportunities to develop new and, perhaps, innovative solutions. IT may be adding capabilities that allow new user features, improved data, or more ability to build in business intelligence. Business may be ready to make significant business process changes that suggest faster, more productive user experience possibilities. Users may bring a paradigm-changing perspective to how the application could be structured. The best Team Leader sees and then champions the optimum mix of possibilities in these three areas of opportunity.

  > The best Team Leaders are able to process a complex mix of input and arrive at optimum decisions. It requires as much, or perhaps more, art than science. As with Business Analysts, and to some extent UI Developers, we noticed that the best Team Leaders are able to maintain a global perspective of the application while at the same time be aware at a very detailed level.

c. Envisions, articulates, and evangelizes the core application user strategy.

- The best Team Leaders nail user strategy. The best applications have a coherent and consistent user strategy that ties the whole application together for the user. It is the structure, the theme, the paradigm which informs the entire application. (See User Strategy Development)
- The best Team Leaders win everyone over to a single, cohesive user strategy. Business, IT, and User Representatives all need to "buy in" to one user strategy. If not, the application development project tends to pull in several directions at once.
- Applications rarely stand alone. They are more often a part of a broader information system the company uses to support their employees and business processes. The best Team Leaders are able to conceive of user strategies that will mesh in well or complement existing user strategies in other applications, or win the various application stakeholders to a new paradigm for future development projects to follow.
- Once the user strategy is decided upon, the best Team Leaders keep it constantly to the forefront during all phases of development, insuring that the user strategy remains clear and undiluted.

d. Deeply involved in the user experience and user interface development process.

- When it comes time for user experience and user interface development, the best Team Leaders have their sleeves rolled up and are right in the midst of the process. (See User Experience and User Interface Development)
- The best Team Leaders lead user experience/wireframe development workshops and often personally create wireframes, or work very closely, intensively, and iteratively with a small group to help create wireframes.

  > This process, where detailed screen designs are actually produced, more than any other single aspect of the development process, when done well, insures highly usable and highly productive applications. This is where the magic happens! And this is where the Team Leaders' enthusiasm and creativity shines through.

Normal Practices

a. The Team Leader is often from the IT side and has more of a programmer's perspective of development than a user-centric perspective.

- The nearly inevitable result is that the Team Leader becomes more concerned about making decisions that will result in efficient code development, or re-use of existing code, than in allowing application development to mirror the users' work and task flow.

  > It is not surprising that the Team Leader has an awareness of the whole project—including the challenges of coding and implementation. It is also not surprising that the Team Leader will be under some pressure to meet their schedule (probably too short to begin with) and stay in their budget (probably too little to begin with). We have yet to participate in or study an application development process where there were not significant limitations and pressures on the team. However, the best Team Leaders manage to resist the easy temptation of reusing or repurposing code and making the user experience fit what they already have. It's always a balance. Time and resources must be considered. But if only time and resources are considered, the user is seldom the winner.

- A strong Business Analyst(s) will help counterbalance this tendency, assuming they are an enthusiastic user advocate.

  > However, if neither the Team Leader, nor the Business Analyst are committed user-centered designers, it will be almost impossible to avoid a data-centered application as a result. Applications often become data-centered because that is the most easily understood and logical way to divide up the functions of the application. Most applications, whether internal, B2B, or even consumer-oriented, allow users to do three things—see information, edit information, or add information. So as an example, it might appear logical to organize an application to see/edit/add information on customers in one area, see/edit/add information on customer orders in another area, see/edit/add shipping information on customers' orders in another area and so on. We call this creating data "buckets." Applications organized this way are very orderly and clear. Unfortunately, they have nothing to do with how users actually work. When an actual employee user is taking an order over the phone they need to see customer information, order information and shipping information all on one screen. This is the core of good application design and is based on task and workflow—not data types.

b. The Team Leader is often required to perform the dual role of Project Manager as well as Team Leader

- When there is high demand on the Team Leader's time (which is nearly always), the strategic and creative demands of the dual role will nearly always give way to the practical demands of running the project efficiently.

  > We see this a lot. It is probably the number one reason why Team Leaders do not provide the kind of leadership required to create a great user-centered application—

he or she is simply too busy managing schedules, chairing meetings, and creating reports for management to have any real band-width left over to be the creative leader of the team.

> When this is the case, user strategy and user experience development are not addressed with much depth (and sometimes not at all).

> A strong Business Analyst(s) can help counterbalance this deficiency.

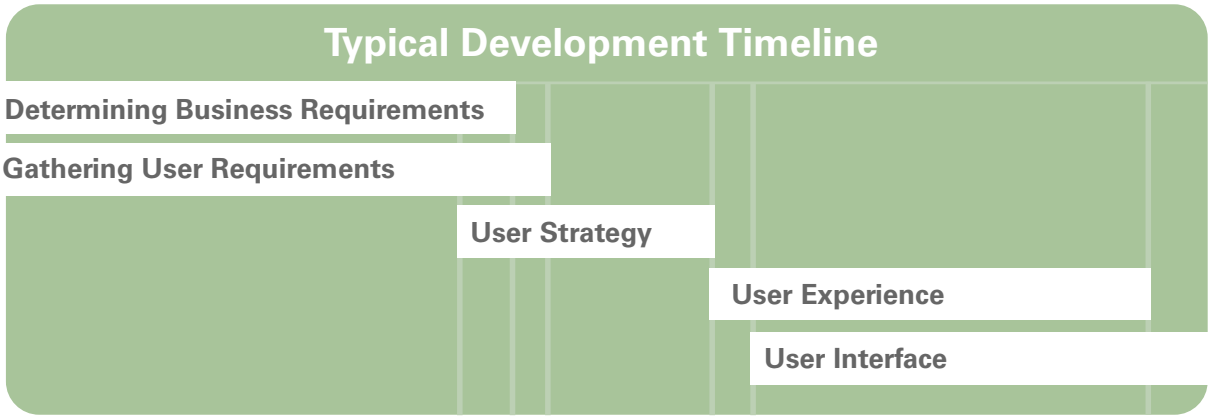
c. The Team Leader sees his or her role as more of a coordinator of other talented team members than the lead creative force of the team.

- This creates some uncertainty about who else, since the Team Leader isn't, should step forward and take a strong role in user strategy and user experience design. Often that uncertainty results in no one stepping forward.

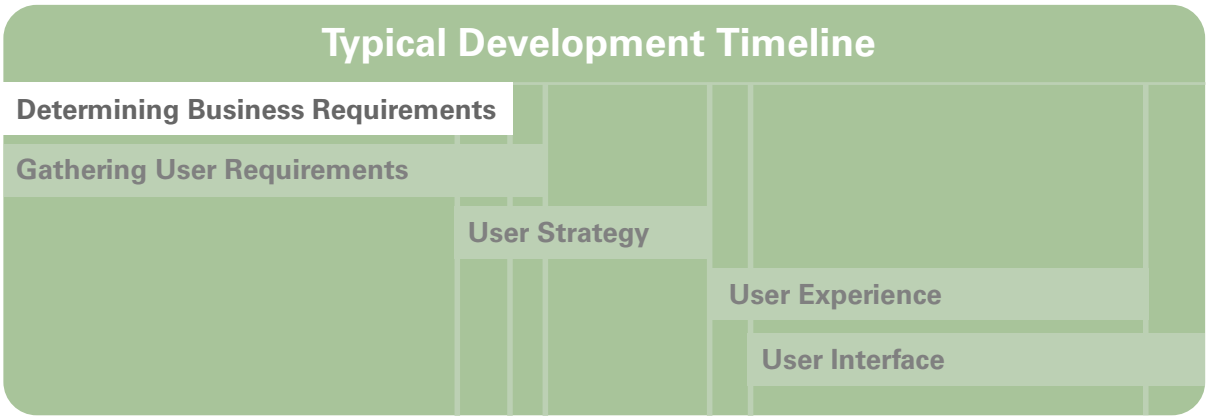
Timeline of Development Phases

In this report we break the development process down into functional phases and processes. We thought it would be helpful for you to see at a glance their relative relationship in a typical timeline.

Note: As mentioned earlier in the executive summary, the best practices we studied and identify in this report focus on core team practices. The typical timeline below applies only to one core team's activities. In application development projects where there are multiple core teams working on a single large application, the phases and processes would be significantly more complex, including overlapping or paralleled core team development phases and processes.



Determining Business Requirements



The most successful application development projects, measured by increases in productivity and user satisfaction, begin with the superior determination of business requirements.

Business requirements are not simply sitting there waiting to be gathered. Business requirements are the result of decisions to improve business processes both at a strategic level (sometimes called program level) and at an operational level (specific aspects of operations). Determining strategic level business requirements involves every segment of the company—from marketing and sales through operations, to IT. This study does not try to address the best practices by which companies determine their strategic business goals; strategic business planning goes well beyond the scope of this study. However, it is important to note that when the strategic level goals are clear and accurate, the accuracy and clarity of operational-level goals is much greater as well, and the entire development process is subsequently enhanced.

Gathering and determining business requirements for internal applications offers a significant opportunity for management to improve operations and thus increase productivity. As such, the more methodical and complete the involvement by management with the application development team, the more likely the results will bring significant returns on investment through increased capabilities and productivity.

Best Practices for Determining Business Requirements

1. Take the opportunity to improve and streamline business processes for the user.

a. The best teams realize that while determining business requirements for the application, the application team is in a position to help the Stakeholder Champion, and other business representatives, streamline or improve the very business processes the application will support.

- When not conducted at the same time as an application development process, efforts to streamline or improve business processes often prove highly ineffective simply because the applications available to the users to complete processes are frequently, de facto, the business process itself. Without a change in the application, there is no change in business process—the application is the business process.

  > Legacy applications, which can sometimes be quite old, often dictate a very

inefficient user workflow, simply because, as information workers, there is no other way they can perform the tasks they need to do outside of the application(s).

b. Savvy businesses realize that new application development and new business process development are one and the same process.

2. Perform work- and task-flow analysis with actual end users, managers, and stakeholders.

a. The best teams bring together the core team, users, and key business representatives with enough time to map existing work and task flow, explore potential ways to improve the work and task flow and arrive at decisions (this often requires multiple days). The nature of the sessions is intensive and highly iterative, and saves enormous amounts of the calendar time usually required to process the same decisions outside of a process—intensive series of group meetings.

- Key contributors:
  - > Users often have insight into work and task flow that their managers do not have, due to the fact that users are minutely aware of the day-to-day reality of the business process.
  - > Managers anchor the business process, and therefore can oftentimes speak authoritatively about what can and cannot be streamlined or improved, and what new changes to the business process will be required in future.
  - > Database Analysts and Programming Leads have insight into what processes can be done in the background or by the system to eliminate steps from the user work and task flow, as well as discuss what data can or cannot be delivered to the application.
  - > Stakeholder Champions are able to seize opportunities for business process changes that arise in the sessions which no one else present could suggest or authorize.

b. Work and task flow is examined user role by user role. The best applications support the work and task flow of specific roles users play within a business process—not the overall business process itself.

3. Capture priority from a business point of view and frequency of use from a user point of view.

a. The best teams create applications which satisfy all business requirements. However, the requirements are satisfied—from the perspective of the user experience and user interface design-in order of their importance and in order of their frequency of use.

- Frequently executed work- and task-flow processes, which have a high business importance, should be most prominent on any screen required to complete them.
- High priority and frequently performed work- and task-flow processes should dominate the formation of the user strategy and user experience.
- Work- and task-flow processes which are less important and/or performed less often should, correspondingly, receive less priority on the screen.
- A well-processed and prioritized set of business requirements also facilitate the inevitable decisions that come in an application development process when there are too many features

and functions to be implemented within time and budget. Some things simply have to be dropped and scheduled for the next release. When priorities are clear, these decisions can be made more quickly and with minimum disruption to the overall process.

4. Iterate with upper management, stakeholder(s), Business Analyst(s), operational managers, Team Leader and users until requirements are finalized.

a. The best teams realize the value of a thorough, iterative process to make the final determination of business requirements and priorities.

- Typically, initial business requirements lists identify significantly more needs and issues than the application can address. The best application teams make sure they are working with a finalized list of business requirements that are within scope and which meets the highest business priorities.
- The best Stakeholder Champions, Team Leaders, and Business Analysts make sure the appropriate amount of attention is received from management.

Normal Practices

a. There is little commitment from business managers whose business processes are supported by the application under development to examine and improve their business processes.

- Frequently managers do not have the time available to examine the business processes for which they are responsible when the application development process requires it. The managers do their best to cooperate with the application team but can only rarely do more than communicate the most pressing problems within the current application, or set of applications his or her team uses.

> The business requirements that emerge in these situations are typically an unstructured mass of requests from business and users. The list is compiled and then thrown over the cubical wall to the development team to do their best with it.

b. Work- and task-flow analysis is non-existent.

- Applications which are designed without work and task flow as their informing paradigm often end up as little more than databases with a web interface.

> This becomes what we call the data "bucket" application. Information the users require to do their work is available to them by type of information (e.g. customer info, vendor info, etc.), often requiring them to "click around" to a number of different screens to find all the information they need, as opposed to providing the right information at the right time on a workflow-oriented screen.

c. Work- and task-flow analysis, when there is any, is conducted by the Business Analyst(s) working independently from the team and key business representatives.

- The Business Analyst often works primarily with users to examine work and task flow, and does not have the advantage of a group workshop where informed and authoritative voices can help reshape the workflow. The end result is that most Business Analysts, finding themselves in this position, can only document the existing work and task flow and have very little ability to change or influence the work and task flow for the better.

> The new application is then built around the old work and task flow.

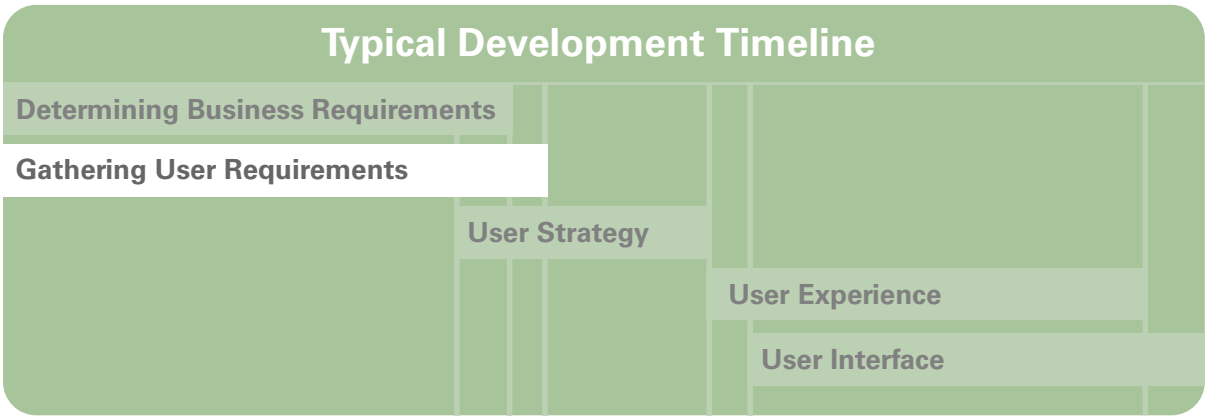
d. Business requirements are collected and approved, but not prioritized.

- When there is no clear sense of priority conveyed in the documentation, everything becomes equally important.
  - > Screen designs often become dominated by complicated processes that are a low business priority and/or used very infrequently by users. Sometimes the complexity of the process, even when a low priority, takes up large amounts of the development teams' time and attention and often overshadows other more important, more frequently performed tasks, as well as dominating screen design.
    - - The lack of prioritization is the most common failing we encounter. It is not that there is no awareness of high-level priorities, but there is no clear prioritization at the detail-level of the application. This often masks differences of opinion that come out during the detailed screen review phase, and which burn up endless hours and countless revisions over issues that should have been addressed much earlier in the process. As the saying goes, the devil is in the details, and that saying is never more true than in application design.

e. Business requirements are often a list of suggested fixes to existing problems.

- Business requirements are often built up from change requests, lists of help desk complaints, and othe sources, such as user surveys, that tend to identify and capture the shortcomings, gaps, and problem areas of an existing application.
  - > This is appropriate for an incremental change process between major releases.
  - > However, when a major new release or an altogether new application is to be developed, the "fixes to problems" mindset works against new user strategy, user experience, and user interface ideas. Many problems are "solved" in new releases by creating new ways that the application can support newly streamlined and improved business processes—the existing problem simply no longer exists.

Gathering User Requirements



Any great (or even good) application design is based on the real needs of its actual users. The best teams engage in a methodical user requirements-gathering process to learn and immerse themselves in those needs.

1. The team sets clear objectives for gathering user requirements, including timelines and success metrics.

Best Practices

a. The team objectives for gathering user requirements are simple and clear, and they include specifics for how the team will include the information gathered into the design process, including the use cases.

- This need not be complicated, just clear. Break your users down into groups and sub-groups, figure out how many of each you can get input from, and how you are going to get that input. Some teams put off user research simply because it seems like a "big deal."
- Where there is a need for extensive and thorough user research, setting objectives becomes highly important. User input on very specific aspects of the application under development, or on very specific task- and workflows, takes careful planning.

b. The objectives and success metrics are understood and agreed upon by the entire team before the user-gathering requirements process begins.

c. The team decides on a practical documentation system to facilitate the user requirements being included in all phases of the application design process.

- Simple is best. Documentation that is performed by team members should include their own analysis. Passing on recordings, transcriptions, or raw survey data is time-consuming for the team (which generally means most other team members will not take the time) and does not give the team the benefits of the team members' impressions and insights gained while conducting the research. Separate the wheat from the chaff.

d. The project timeline provides adequate time for the gathering, documentation, and prioritization process.

- Many teams complain that they do not have time for user research, when in fact they never even try to get it in their schedule. The best teams make the time by working it into their schedule at the very beginning of the project—even if all they can spare is the time to talk informally with a handful of users.

Normal Practices

a. Teams may have an orientation toward the importance of user requirements input into the design process, but have no real plan for how to achieve it, other than doing "acceptance-testing" toward the end of the development process. Generally, once the development process has reached the stage of acceptance-testing, it's not feasible for serious user experience modifications to be made.

- There is an unstated confusion on many teams between user research and user-testing. User research, generally speaking, is a more informal process consisting of interviews, asking users to perform scenarios, exploring task- and workflow in workshops and doing a lot of watching and listening. This kind of user input allows teams to get a feel for the users and their most important concerns, needs, and problems. Without this, it is hard to begin designing even the first screen of the application. Once screens get designed, then user-testing becomes appropriate in order to refine and improve on the core experience design. But if your core experience design was not informed by prior user research, no amount of user-testing is going to help you overcome core user experience issues—short of starting over.

b. Teams may do some initial user requirements-gathering, but have made no plan in terms of process or documentation to include the information into the application design.

- User needs, like business requirements, need to be distilled and prioritized. If not distilled into specific features and functions, the user needs remain as a generalized concept which can not get inserted methodically into the process. If user needs are not prioritized, then teams have to deal with the same problem as non-prioritized business requirements—everything is of equal importance and features which are, in reality, of lesser importance can end up dominating screen designs.

c. The timeline doesn't allow information from the user requirements-gathering to actually influence use cases and/or interface designs.

- Gathering user requirements, gathering and determining business requirements, and clarifying infrastructure and programming parameters should all be done in parallel and come to conclusion before any of the design process begins. Once use cases are written and/or screens are being designed, it is very difficult to get significant and new features added and therefore new features are added only when they have a very high priority. Realistically speaking, given the schedule pressures of most projects, information not known before the development of wireframes and use cases will not make it into the application.

2. The user requirements-gathering process begins with well-defined user segments—those core user group or role types who will need unique features and activities in the application interface. (In the Designing User Experience section you will find more on this subject.)

Best Practices

a. The best teams are aware of the importance of going into user strategy and user experience phases with a deep understanding of the distinct user tracks the application needs to support. Without this, the application becomes too generic, and forces various user types to alter and adjust their natural and productive workflows to the interface.

- This is a puzzling challenge for most development teams. If all things were equal, the team could, and would, come up with a fairly large number of identifiable user groups. However, since most teams are already bound by a finite development schedule and budget, they are uncomfortable coming up with too many distinct user groups and therefore introducing what seems like too much complexity into the design. Good Team Leaders embrace this process, but in the end have to limit the number of user types that will be planned for, and supported, in the application. Going through this process can, however, be very revealing. Often there are fewer user types than would at first appear, because even though there are many different roles, with different titles expected to use the application, on examination teams discover that their needs are similar or that minor additions or deletions of features or screens can suffice to customize the application for another user type.

b. Many teams at this point use the methodology of creating "personas," or archetypal users to make each of the segments or roles more real for the designers.

- It has been our general observation, with exceptions, that the development of personas is best suited to distinguishing between users of publicly accessible sites. In this realm, users can choose to go to other sites. In this realm, designing a site to satisfy not only functional needs but more subtle personal and emotional needs of users becomes essential in a competitive market. In internal applications, breaking users down by functional roles is done more straightforwardly by examining business processes than by developing personas.

Normal Practices

a. Typically, teams design too generically, for a perceived "conglomerate" of the various user segments. Or they design for commonly accepted user types that are actually based on typical organization divisions of the company, such as marketing and accounting, etc., which do not translate to any specific user group. The generic interface doesn't really serve any of the segments well.

3. User requirements are identified early on in the project, at the same time or before business and functional requirements are developed.

Best Practices

a. Development teams conduct, document, and prioritize user requirements before use cases are developed, and in parallel with or before business and functional requirements-gathering.

Normal Practices

a. Teams engage in user requirements-gathering after business requirements documents and use cases have been developed and even signed-off. At this point, because of timeline and project scope limitations, the information cannot be refined and integrated.

4. Team members who will later develop user strategies, user experience designs, and/or user interfaces for the application participate directly in gathering and developing user requirements.

Best Practices

a. A core team conducts and documents the user requirements. These same people are directly involved with developing the user strategies and user experience conventions, the visual design of screens, such as wireframes, and initial interface designs and prototypes.

- These team players' direct involvement with users tends to influence them to design interfaces more resonant with users' actual work and task flows. Ideally, but not necessarily, these are trained user experience designers.

> Can't say enough about this point. Essentially, there is no substitute for direct contact and involvement with users. Even well-documented user research lacks most of the flavor you get when you have direct contact. The old saying "You can't really know someone until you walk a mile in his or her shoes," is very appropriate here. If the team members doing the actual user experience design have not walked a mile in their users' shoes, they will always be off in their understanding.

Normal Practices

a. User requirements are gathered by a separate team, individuals, or a third party organization, who pass this information on to those involved with designing and prototyping the application interface via documents.

- We do not mean to imply here that core team members must do absolutely all their own user research. Research performed by others, or outside vendors, and passed on through effective documentation and presentation, can be highly valuable and influential, but it cannot substitute entirely for the core team members having direct user interactions.
- One of the development team members we interviewed for this study made a point of going to visit users of his application once a quarter. His visits weren't marked by highly scripted user research, but he found his informal exposure to the users to be highly valuable and is attempting to do it more often.

b. User requirements are gathered and documented solely by business team members, or by IT team members, who are not involved later on in the development process with developing user strategies or user interfaces. All input is passed via documentation.

c. User requirements are gathered by teams or individuals who are not actually trained and/or interested in the value of user input for excellent application design.

- We have encountered a fair amount of lifeless user research. We use the term "lifeless" to describe the documentation because the results are presented in such a way as to remove any of the human feel from the research. Numerical results are often this way. Surveys are often this way. Good researchers know how to pass on the intangible and significant impressions they have gained along with the facts. It sounds odd to say this, but sometimes the numerical survey results which indicate a user really wants something, or really dislikes something, simply aren't true. It is more a matter of how the question was phrased, or a matter of timing. Time spent with just a few users going over survey results can make the dry facts come alive.

5. User requirements are treated uniquely and distinctly, and are not simply a sub-set of business requirements, functional requirements, content requirements, etc.

Best Practices

a. User requirements, as gathered from actual users, are considered separately from other application requirement processes, such as business requirements, business directed content requirements, and IT team directed functional requirements.

- The best teams translate their user research into specific user features, functions, and activities that their user research subjects have requested. Like business requirements, these user requirements are prioritized. User involvement in the prioritization process is encouraged but can get cumbersome if too many users are involved. A simple high, medium, and low pri oritization will usually suffice.
  - > Without this process, most user requirements get lost. With this process, user requirements make it into a features and activities list that combines user and business requirements and which will be included in the design of a screen or screens.

b. In the hierarchy of requirements leading to user experience interface design, user requirements have a predominant position.

- The best teams become user champions. Business interests are well represented in most development projects, as are ITs' needs and concerns. Users rarely have a seat at the table, or if they do, they do not have anywhere near the authority and influence of the other players—unless the team consciously puts their needs forward.

Normal Practices

a. User requirements are included in other processes of requirements-gathering—be they business requirements, functional requirements, and/or business-directed context requirements.

- We were often told that teams had gathered business and user requirements only to discover that the source of both was the same—managers, supervisors, and other players whose primary focus is business processes and operational issues. In their minds, there was no difference between user needs and business needs.

b. User requirements are considered secondary, when strategically compared to other requirements, such as stakeholder-directed business requirements and IT-directed functional requirements.

- This attitude is unfortunate for two reasons. One, and the obvious one, the users' needs do not get met. The second reason, however, is that it is unfortunate because in many instances there is no conflict between businesses priorities and user requirements. Often the business requirement is "what" and the user requirement is "how." But discussions about how things should be done in the application are frequently overshadowed by discussions of what should be done. The two are not mutually exclusive but an overly strong business focus tends to crowd out user concerns.

6. The user requirements-gathering process is based on interacting with actual users who represent core application user types.

Best Practices

a. The user requirements-gathering process is done with current, actual users (or typical potential users) of the application.

b. All of the major user "types" and demographics are represented in the requirements-gathering process (workers, managers, stakeholders, executives, first time, frequent, infrequent, male, female, young, old, experienced, inexperienced, etc.)

Normal Practices

a. Teams or individuals will sit in conference rooms and create user requirements lists as a logical extrapolation, isolated from actual users.

b. Requirements are gathered by development team members meeting with managers of the various user types, rather than with the people who actual do (or will) use the application daily as their primary work tool.

c. Teams will often gather information from a limited set of users who do not represent the full range of current and future user types. The resulting application design often does not fully serve each of the different kind of core users.

7. The gathering process includes "contextual" interviews with actual users (one user at a time, at their place of work).

Best Practices

a. The gathering process is based on "contextual" interviews (one user at a time, at their place of work). Users are not only interviewed but observed at their workstations, doing the core tasks that make up their workflow.

- We have observed that it is in a contextual interview only that team members really learn what is missing from their application. As you observe users performing typical tasks or give them scenarios to perform, you will notice how often they must "go out of the application" to get their work done. We frequently see users consult binders, papers pulled from desk drawers, other databases, their email, etc. in order to complete a task. Many of these other sources of information can and should be incorporated into the application under development.

b. The best teams create a safe environment for corporate users who participate in the requirements-gathering process, respecting their privacy and their relationships with superiors, co-workers, and those in the company responsible for creating and maintaining the application.

- Typically it is best not to have the users' immediate supervisor present during the interview! It is almost irresistible for the user to say what they think the supervisor will want to hear.

Normal Practices

a. Teams send out written or online surveys to collect requirements.

b. Teams conduct focus groups, where users report what they do or need to do. The resulting information is limited by many factors. Users will often report that a task in an application "works fine," but when actually observed performing that task it is found to be inefficient and error-prone, etc.

c. Users are often reticent to speak in group settings with managers and superiors present, or concerned about making negative reports about an application they know has been created by others in the room.

8. User requirements documentation is useful, minimal, feature-oriented, user type-categorized and referred to consistently throughout the development process.

Best Practices

a. A prioritized matrix of all the major user types and their demographics is created before conducting interviews with users. All subsequent documentation is created in such a way that the data can be viewed by user type categories (whether spreadsheets or databases).

b. In the best teams, documentation is done by the people who conducted the user interviews or group sessions users, and done soon after the meetings.

c. The best teams often create two kinds of documentation, one more textual (or diagrammatic), which is rich with work- and task-flow information, the other more feature-oriented (such as a "Feature and Activity Matrix").

d. Great teams often re-read user interview documentation before or during group meetings and work shops to keep the user orientation in the forefront of their design process.

- The best teams all showed a ready facility to produce user documentation when needed in meetings. These documents are referred to as commonly as business requirements.

e. Good documentation helps the best teams to never lose important user information. The documents are designed to serve as prioritization tools, and roadmaps for the evolution of the application tool.

f. The documentation is read and assimilated by all members of the team, including stakeholders, is used as a primary interface design tool, and is re-visited and referred to throughout the life of the development project.

Normal Practices

a. Normal development teams usually tend to create one of two types of documentation, either a textual transcript of the interviews, which tends to be unwieldy and difficult to extract practical information from, or an oversimplified list of requirements (which doesn't retain any user work- and task-flow information).

b. A separate group in the corporation, or an outsourced team, will conduct the user interviews and do the documentation. Often those directly involved with design workshops, task-modeling, and even creating wireframes and interface designs have never read it.

c. Teams usually read the user requirements documentation once, often superficially, distill some information from it into other document tools (such as business requirement documents and use cases), and never refer to it again.

d. Teams rarely refer back to original user research documentation in future phases of an application's life, assuming that the information only applies to the original release. This is mainly because the information has not been documented in such a way that it can be used as a road map (with requirements prioritized and organized into current and future phases).

e. Most often a small subset or even one individual in a development team is tasked with reading user requirement documentation.

9. The methodology for gathering and documenting user requirements is appropriate to the project.

Best Practices

a. The best teams understand the importance of gathering and documenting user requirements, but also work within their budgets and timelines.

b. Good stakeholders and teams understand that there are more and less formal (and hence, expensive and/or time-consuming, and labor-intensive) methods of gathering user requirements and usability information (such as Usability Labs with recording equipment, etc). A good example of effective but less formal and expensive methodologies is Jakob Nielsen's Discount Usability Engineering, which includes user research not conducted in formal labs.

- One of the surprising statistics we gleaned from Nielsen's research is that a team can learn up to 80% of all they can possibly learn from users by interviewing 6 to 8 users. The best teams do not over-research.

c. In the best teams, the methodology chosen is considered by all on the team to be appropriate to the scope of the project and cost, time, and labor-effective.

d. Good teams recognize that even a small amount of time spent in direct user requirements-gathering, even informal, even conducted by people on their team not formally trained—will result in a superior application design.

- Sometimes all there is time for are a couple of ad hoc user interviews. It is perfectly appro priate to do so and much better than no direct contact at all.

Normal Practices

a. Typical teams do not distinguish between gathering user requirements, business requirements, and functional requirements—and therefore do not use an appropriate and distinct methodology and documentation system appropriate for each.

b. Many stakeholders and development teams balk at engaging in a specific user requirements-gathering process because they are convinced that the only truly effective methods are expensive, time-consuming, or beyond their group's skill set.

c. Team players often feel that the only valuable data is that produced by highly formalized and scientifically verifiable or numerically benchmarked processes. They don't understand how informative less formal or less extensive methodologies can be.

- This is probably the single biggest point of resistance teams have to doing user research. Use research appears daunting, and they don't think they have the skills or experience for it. Some of the best applications we studied were developed by teams that had had no previous training or experience in user research. They just dived in. They certainly could have done it better had they had experience, but their simply doing it as well as they could helped significantly with their user satisfaction results.

10. User requirements are analyzed and prioritized by the entire team (or by key representatives of each role group in the development team).

Best Practices

a. In the best teams, time is spent in analyzing, weighting, and prioritizing the collected user requirements.

- It is hard to overemphasize the importance of this. Most teams we observed spent a significant amount of time studying and discussing business requirements, but only the best teams spent a comparable amount of time studying and discussing user requirements.

b. Moreover, that weighting and prioritization comes from three major perspectives, as represented by different player roles in the team: business, IT, and user.

c. The best teams have a system for firmly deciding what user-required features, functions, and activities will be included in the current phase of the project before use cases are formalized and actual interface screens begin to be designed.

d. The prioritization includes a roadmap for forwarding requirements into future application phases.

Normal Practices

a. Typically, a list of user requirements is generated but is not weighted or sufficiently prioritized, especially from the business or user perspectives.

b. The design process often becomes one of taking the overall list of user requirements, attempting to design them all in, and finally after a long process of meetings, workshops, document generation, and individual document processing-dumping a great deal of them because they are determined (usually from the IT perspective) to be "out of scope" for the current project.

- Most often this means untold hours have been spent documenting requirements and features (in use cases, in spreadsheets, etc.) and actually designing them into navigation strategies, wireframes, and screen prototypes—all that will have to be reiterated.

c. Many teams make little or no attempt to cull out user requirements and features that have been deemed out of scope for the current project and save them in some sort of formal way for future development.

11. User requirements include How-it-is-now-done and How-it-really-ought-to-be-done work- and task-flow analysis.

Best Practices

a. The best teams recognize that for an application to truly increase efficiency, productivity, and quality of service, it has to be designed to and optimized for the users' optimal work and task flow.

b. In these teams, work- and task-flow analysis is done as an essential part of the development process, and it includes understanding users' current way of working, their optimal way of working, and the business-required evolution of their workflow.

- For applications that are the employees' main work tool, doing this well produces the highest level of return on investment for the company through the resulting increase in productivity. Many employees in the corporate world are information workers. They depend on applications to do their job. It isn't that the application is the tool to do their work—the application is their work. When applications support an optimized, streamlined, and efficient number of steps for workers to get their jobs done, everybody wins. The company gets increased productivity, users have highly usable applications, and the end results include higher accuracy and customer satisfaction. The best teams work hard to get this right through truly collaborative meetings with users and business decision-makers.

c. The best teams understand that applications need to be designed with a conscious and inherent flexibility, so that as the company grows and evolves, it can be easily modified (on a core application and back-end architecture level, a modular construction level, or a user customization and/or personalization level).

Normal Practices

a. Corporate applications are most often fundamentally architected around "data types" and not around core user types' daily work and task flow. Data-centric navigation and application architectures often force users to work in unnatural, and more critically, unproductive ways. Also, service quality is often compromised (as in "call-center" or "customer-support"-based tools).

- Data-centric architecture is convenient for business and IT, and eases the process of development.

b. In corporations, we often find that once an application is created, because of the time, effort, and money involved, the application tends to "drive" the way people work—even when highly unproductive. Once established in the structure of the application, it will become very difficult to refine or evolve those workflows.

- We have seen in work- and task-flow workshops the "evolution" of application-driven work flow through several applications or versions of applications. There have been some ruefully amusing surprises for the development team and business leads. In one case, our primary group of users was required to handoff an initial customer set up process to yet another group, which in turn used another application to do the set up. Our users found this very

frustrating, and it often caused delays in customer service. Once we were able to get all the people together who managed the process, we discovered that 90% of the process that was handed off no longer needed to be handed off—and really hadn't been necessary for some time. Discovering this was a huge win for the users and led to a much more efficient work process.

c. Typical companies resist the money and effort it takes to design flexible applications which can be easily modified for evolving work and task flows.

12. The application development organization has a well-developed, ongoing user requirements-gathering process.

Best Practices

a. The best teams have an ongoing user requirements-gathering process. They have documentation tools—databases, spreadsheets, etc.—that allows easy information input and effective retrieval.

b. The best teams engage in ongoing user interviews, contextual usability studies, engaging actual users in ongoing requirements-gathering, ongoing product-testing, etc.

Normal Practices

a. Typical teams have a minimal ongoing user requirements-gathering process which generally includes huge and unwieldy spreadsheet summaries of help desk requests, emailed feature requests from users, etc. This data is often not summarized or presented in a form that the development team can easily integrate into later interface design phases.

b. These teams have to ramp up a new user requirements process each time they create a new application product. This process becomes a challenge to the team in terms of timeline, budget, and personnel resources.

c. Teams rarely engage in ongoing user interviewing and testing, perhaps doing minimal research such as this only in the initial phases of certain large and important application projects.

13. User requirements are gathered by team members who are trained, experienced, and skilled in user experience design methodologies.

Best Practices

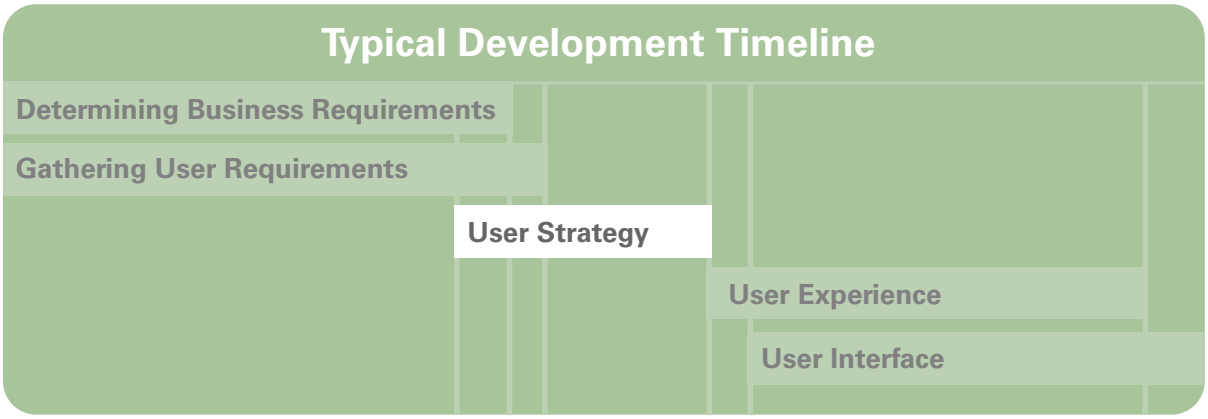
a. Even though, as mentioned above, good teams undertake user requirements research no matter what their budgets and personnel resources, the best teams, either internally or through outsourcing, engage skilled user research and usability specialists.

.

Normal Practices

a. Typical teams may not even be aware of the importance of methodical user requirements-gathering. Those teams that are aware, when faced with the realities of budget and timelines, often drop or minimize the importance of skilled requirements-gathering. They feel it to be important, but not critical to application development.

Designing User Strategy



The word "strategy" means many things to many people. In his book Strategy Bites Back, Henry Mintzberg characterizes a "strategy" as being a plan, a ploy, a pattern, a position, and a perspective. What we mean is basically a unique perspective and plan.

In order to create:

- An effective and productive application,
- A consistent, easy-to-use interface that doesn't require extensive training and help-desk support,
- An application that supports business requirements and the overall corporate strategy,

development teams need to have a definite plan, an interactive paradigm for each of the application's user types.

Without clearly defined user strategies as a continuous informing reference point, it is impossible for development teams to make the literally thousands of decisions needed to create a consistent and user-centered application.

User strategy, when not clearly and specifically defined and formulated, is present in the process, but embedded in documents such as business requirements documents and use cases and does not inform the development process to the high degree that it should. Moreover, these strategies have often been developed ad hoc, seat-of-the-pants, in-real-time conference room meetings as a by-product of other decisions being made, and do not have an overall cohesion or consistency.

Teams that design the best applications define their user strategies before making detailed decisions. Simple strategies, by user type, can then serve as benchmarks and decision guidelines, greatly streamlining the group decision-making process, and more importantly, facilitate the design process toward the desired consistency, ease-of-use, and productivity that all great applications exhibit.

These strategies needn't be bulky documents. A one-page document with 4-8 bullet points for a core user type is all that is needed.

> At a Fortune 100 company interviewed for this report, a core application suffered from repeated and similar user complaints, and also a fairly low score in the application rating. The problem was that the

application had two fundamental types of users, infrequent and heavy. Literally thousands of users would come into the application only once or twice a year; others would spend weeks or months in the appli cation on a daily basis.

But the interface was identical for each of the two user types. If the company had simply started out with a clearly defined strategy for each user type, it would have solved the whole problem and made user productivity significantly greater.

It could have been no more complicated than stating in a very brief document that there would be two major interface paths in the application. One a bullet-proof, single-screen, step-by-step wizard help message-oriented experience for the infrequent user; the other, a more data-dense, robust enterprise tool for the frequent user.

1. The best teams make development of user strategy an actual development process step, and perform this step in the beginning phases of development so the strategy can actually inform the development process.

- Only the very best teams could tell us quickly and easily what the user strategy was for their application. They may not have used the phrase "user strategy," but once they understood what we were asking about, they could quickly describe their overall strategic approach. They might say some thing like, *"We wanted our users to have a decision-tree experience, like TurboTax's online version, where you absolutely can't make a mistake and context help is always just one click away."* Or perhaps, *"The bulk of our users are on the system eight hours a day. They hate it when there is too much help and too many pages. We made the system really robust, so the power users could move at max speed."* Or *"We had to make two tracks—an internal user and an end user. We just couldn't make one common interface work for both. The look and feel is the same, and the basic steps required are the same, but we had to make them two different experiences on a page-by-page basis."*

2. User strategies flow from teams with a strong, current knowledge base of user research, interviews, user requirements, and work- and task-flow analysis.

Best Practices

a. The best teams begin the process of developing user strategies by reviewing and immersing themselves in current user research and requirements.

b. The best teams regularly involve users from all the major user types in the process. If not members of the design team, they will invite User Representatives to meetings and workshops to confirm their findings surrounding a user type's unique needs, requirements, and work and task flows.

c. The best teams will engage in and document work- and task-flow analysis with specific user types before developing an interface strategy for them.

Normal Practices

a. Quite often development teams will take existing artifacts of user requirement data— spreadsheets and databases of user statistics, help-desk requests, past feature and functionality requests, etc.—

and immediately assign a couple of team members (such as Business Analysts) to proceed immediately into use case development.

- Even if new and extensive user research is underway (often even work- and task-flow analysis), these teams often consider that no new information significant enough to radically alter the use cases will appear. Or they feel that the user research results will be used to confirm their work and "fill in the small details" later on.

b. Typically, development teams do not document user strategy separately, but will embed it in use cases or other documentation. This means that in actual practice, the team members tasked with writing the use cases develop the strategy.

- Teams tend to discover too late how bound they are to a user strategy that is created this way. Use case writers have to make certain assumptions, even though (in theory) use cases do not imply interface design or user experience design, they do tend to lock teams into a work- and task-flow paradigm—which is often simply the existing work and task flow which is dictated by the current application.

c. Teams often do not have users read and confirm research they have done on that user type, or review strategies they have developed for them, such as found in use cases, until the project has proceeded to prototyping.

3. User strategies are initially created from the ideal user experience perspective. Business requirements, IT programming concerns, scope issues, timeline issues, and budgetary considerations will be critical to make the strategies realistic, but they shouldn't dominate initially.

Best Practices

a. The best teams see the user strategy development process as their main opportunity in the project for real vision and innovation.

Some person or persons had the foresight to imagine an accountant being able to harness the power of a room-sized computer in a small box and perform real-time, interdependent, multi-sheet, highly complex accounting tasks on a single screen.

Hence, a primary chapter in the desktop computer revolution begins.

This is a great user strategy.

b. Great Team Leaders guide their entire team into a process of unrestrained, out-of-the-box, brainstorming of how to fulfill the users' (their employees') goals through their work and task flows, while at the same fulfilling business requirements and strategies.

- These teams know that even if parts of an ideal user experience may turn out to be impossible or financially unreasonable from a programming perspective, or out of scope for the current project phase—there may very well be other ways to "detail" out and design to the ideal.

c. After the initial ideal user experiences are identified, the best teams go through a process of "making them real," and also identifying aspects of the user experience to shift to future phases of the project.

- When you are a decision-maker charged with managing the business outcome of a development project, these sessions can feel a little overwhelming or even threatening. The best teams quickly move through brainstorming to grounding the best ideas in practical doable designs. But without the opportunity to think out of the box—even for a short while—applications often end up being just improved echoes of the previous application.

Normal Practices

a. Typically there is a fair amount of division in teams about spending time discussing (or worse, documenting) the ideal user experience. Many feel it to be a waste of time, and will immediately jump into demonstrating how unfeasible any ideas generated are.

- These kind of sessions require a good facilitator. When someone in the room starts suggesting a direct neural link approach—and the facilitator writes it down on the white board—you are probably in trouble.

b. Quite often, a majority of team members come into a development process with disparate basic pictures in their minds (or on paper) of how the application will be architected (both from a data model level and from an interface level).

- The development process becomes one of arguing for position and the resulting decisions are often the least common denominator solutions—the ones that can be agreed upon by the team in a timely manner.

c. Even if teams outline user strategy, either in user strategy documents or use cases, they haven't really gotten full buy in by all the roles in the team (largely because the whole team doesn't understand the full implications of the strategies because they are buried in large documents).

- Because of this, they don't go through a process of making the strategies real, in terms of modifying them to work with important factors such as business requirements, scope, programming feasibility, etc.
- But later, when screens begin to be prototyped and code-formulated, these problems show up, inevitably and expensively.

4. A complete set of user strategies is created, one for at least each primary application user type and sub-type (financial user, manager user, executive user, sales user, frequent user, infrequent user, etc).

Best Practices

a. The best teams base their strategy work from a well-defined matrix of user types and sub-types for which they have done user research and gathered unique requirements.

- The best teams are aware that one of the fundamental pitfalls of application design is to design a one-size-fits-many interface, assuming that one or many user roles do (or can) work just like another.

- Hence, good teams may create significant differences in an effective application interface for even superficially similar roles, or they create a very flexible and dynamically customizable design.
  - > For instance, a team assumption might be that one user strategy may fit every accounting user in an internal enterprise tool (the assumption being the differences will be insignificant and easily worked out on a detail level), when in fact the domestic invoice auditors' workflow differs fundamentally from the international auditors flow.
  - > But the initial interface designs are not flexible enough to customize, and one of the accounting user sub-types is forced to work in an unnatural or unproductive way in the interface.

b. Great development teams consider subtle variations in interface strategy for subcategories of user types. Often there will be an overall application strategy for these situations (such as a dynamic, in-context help system that can be turned on for new and off for experienced users).

Normal Practices

a. Often teams do not have a spreadsheet or database defining the full range of application user types. They jump into use case development for the broadest or most known user type, from a business requirement or functional requirement perspective, and basically develop the user matrix as they develop the use cases.

b. Teams typically design application interfaces that are too generic, assuming one user type works in the same way that another one does. Later, when actual user feedback or help desk statistics come in, the team is forced into costly redesign or quick releases to fix the problem.

c. Teams often base navigation systems on data types, and not on work processes. For instance, they may create separate links in a system for customer profile information, for recent orders, and for order history. But in a standard interaction with a customer, a typical internal user may need to access all three of these data types in a specific call order, all on one screen.

d. Teams assume that obvious and broad corporate user type categories such as accounting, and sales, etc., are sufficient on which to base critical interface detail decisions, such as navigation paradigms. They don't confirm this, and/or strategize what “actual” user types really need interface variations.

e. Typically, teams will develop some kind of application embedded help system, but rarely strategize how this system might be flexible enough to perform differently for different user types.

5. User strategies are determined at a high level, simply documented, kept distinct from any other interface details, and are developed before use cases and/or any interface details are worked out.

Best Practices

a. The best teams create simple, documented, high-level strategy statements that specify the core direction interface details should take for each user type. Only after these strategic decisions are clarified can they move on to the details of the interface.

b. The best teams review and use the strategy as a benchmark and mediator for group decision-making processes.

Normal Practices

a. Typical teams bury their strategies in other development documents, such as business requirements documents and use cases.

b. These teams generally don't develop the strategy first, they begin making individual feature and requirements decisions, and begin to see a strategy develop as a result of those decisions.

6. User strategies are agreed upon by the entire team, and used as benchmarks for every specific user experience and user interface design decision made throughout the project.

Best Practices

a. The best teams use the user strategy development meetings, or workshops, as an opportunity to arrive at a shared understanding and consensus. They involve the whole team, including stakeholders, business, and IT representatives.

b. Teams do not make any interface detail decision that is in conflict with the strategy. In this way they create a consistent user experience and are able to fulfill the vision of the strategy.

Normal Practices

a. Typically, teams will have the Business Analysts or other members develop user strategy and then work on a "respond and revise" basis with other members of the team. This means that many team players can only absorb the strategy by reading long and detailed documents.

- Many team players may hold onto and not communicate fundamental disagreements until the process is too far down the line to adjust to real issues. Other players (such as non-technical roles), including those who don't absorb detailed documentation easily, or who haven't the time to study them deeply, may not even understand the strategy until they first see screen mockups.

b. Without simple, high-level guidelines, teams have no clear benchmark against which to base decisions where there is disagreement within the team. This tends towards a fragmented and inconsistent user experience.

- This happens especially in large enterprise applications, and especially where there are multiple teams or company divisions responsible for various areas of the application.

7. The user strategy is modified and refined as needed in the course of designing the user experience and interface, and throughout the life of the application tool.

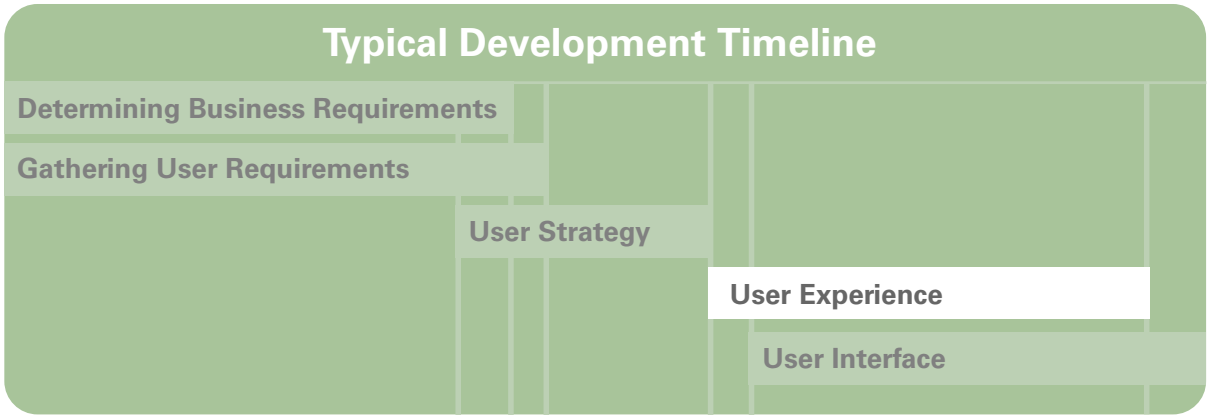
Best Practices

a. In the best teams, if a user experience convention is recognized as being correct and important, yet it counters that user type's strategy, the team modifies the strategy.

Normal Practices

a. Typical teams (who create and define user strategies) see them as a one-time-and-we're-finished type of activity, and not a tool that may need to be refined as the application develops.

Designing the User Experience



The User Experience (UE) is the sum of the dynamic interactions (e.g. module behaviors), conventions (e.g. most recent data at the top of lists) paradigms (e.g. multi-step wizards in pop-up windows), and work- and task-flow steps which users learn in order to use the application. The UE is derived from and is a more complete and refined expression of the user strategy.

User experience, more than any other factor, will determine user satisfaction.

1. First Things First, the Flow of Application Interface Design

To the degree that teams design, detail, and document the user experience before moving on to user interface design, to that degree they will be better and more effectively able to focus on user interface issues when in that phase.

- This is especially important when teams are working on a new release or on incremental changes. Typically, the documentation that most insures continuity of experience design conventions is a robust style guide that includes user experience conventions as well as the usual user interface building blocks.

2. Defining User Types and Subtypes

The best applications are highly responsive to the needs and work and task flow of specific user types. In order to achieve this, teams need to begin their development process with an intimate awareness of the distinctions of each user type. This can be a significant challenge. There may be hundreds of user types in a large organization. The best teams create, update, and maintain documentation describing and defining every user type and subtype the application supports or will support.

In any application design process, decisions must be made about how much to customize interfaces for various user roles, how much personalization and user preference setting to include, whether to have dynamic customization based on log-in or more static customization based on navigation choices and preference settings.

In B2B applications, decisions as to whether to have completely different interfaces for highly distinct user roles (such as outside contractors, service partners, etc.) must be made.

In a B2C application, the process of defining user types and subtypes can be particularly challenging, and the important distinctions generally move from workflow and role-based definitions one finds in B2B, to factors such as demographics, relative computer skills, frequency of use, or connection speeds: Are users male, female, what is their geographic location, are they married, do they have children, are they new users, skilled users, what is their connection and browser type, how big are their screens, etc.? This kind of information is often more easily obtained through Web use statistics-gathering, click-track analysis, online surveys, sales and marketing data, data-mining techniques, etc.

Best Practices

a. The teams that create the best applications are in organizations dedicated to understanding their users, be they completely internal, B2B, or B2C applications. Moreover, they have ongoing processes and tools to collect, prioritize, maintain, and update this information. Generally this takes the form of some variation of a User Matrix, be it a spreadsheet or database.

b. These teams are careful to make this information approachable, in terms of volume or filtering, giving the team the ability to distill and/or summarize the most useful data.

- The best teams create an easily referenced document that contains key descriptions of the user types and subtypes that the application is supporting (or will support if it is a new application). Typically, this is not an elaborate document. A simple spreadsheet is most often used.

c. The best teams collect this kind of information through direct user research, rather than solely through business organization/division information, or through contact with division leaders and managers only.

d. Great teams have User Representatives from many core types who take on the role of gathering and maintaining application needs, such as features, from their coworkers. In the best scenarios, the same User Representatives are also intimately involved in the application design process, attending design workshops, reviewing prototypes early on in the process, etc.

e. Many of the best teams, especially those developing B2C applications, make use of personas. The team creates user personas, which make user groups real to the development team: personalities, demographics, work and task flows, computer-use orientations, etc., that make up their user types. The best teams consistently and effectively use these personas as design tools and benchmarks throughout the development process.

Normal Practices

a. Typical teams will have knowledge, of course, of their user types, and an awareness that the application needs to serve distinct roles—however, they have no ongoing business process or tool to maintain the information, and thus with each application project the users' needs have to be re-created as a part of the project scope.

b. Typical user type information is more business division-oriented, and is not described or defined in detail in terms of important application feature distinctions in work and task flow.

- Where the application serves internal users only, we often find that users are wrongly categorized according to the functional divisions of the company: financial services, sales, customer service, etc. In reality, the needs of a specific user type may span several functional divisions or, and this is more often the case, within one functional division of the company, i.e., the customer service division, where there may be two or more distinctly different user types.

c. Often teams have no User Representatives aiding in collecting and maintaining needs specific to their group (one of the richest sources of actual interface details).

d. Many teams will create personas, and even create them well, but then will move on in their design process and never refer to them again, thus mitigating their value as a powerful design-for-the-user tool.

3. Work- and Task-Flow Analysis

The best teams conduct work- and task-flow analysis early on in the development project whether or not they have a formal methodology. If nothing else, they use informal meetings and/or workshop whiteboard sessions. Whatever the method and documentation, the analysis produces information that is practical and useful. It can be understood by the team and actually brought into the interface design process. The knowledge gained from work- and task-flow analysis, more than any other single type of user information, leads to highly productive applications.

Best Practices

a. The best teams involve a broad group in the process—the core team, stakeholders, and business repre sentatives, such as operational managers, and spend invaluable time hearing and absorbing how real end users specifically do their work.

- It is hard to stress enough the importance of getting the right mix of people to attend these workshops. If you don't have real users present who perform the work day in and day out, you will not catch many crucial steps to work processes, or not have a realistic sense of sequence of the tasks required. If you do not have business process owners represented, golden opportunities to streamline work processes will go unrealized. Many companies resist providing these resources to two, three, or more days of workshops because they are considered to be essential elsewhere, but the end results more than compensate for their time.

b. The best teams not only analyze current work and task flow, but spend time innovating and analyzing new and optimal flows. Substantial productivity gains come from business process changes discovered and championed in the process of analyzing work and task flow of specific user types.

- The best teams realize that this analysis will implicate changes in even core business and operational processes and practices, and they are willing to go up the chain of decision-makers to fight for these changes as part of the application development process.

- We have even participated in work- and task-flow workshops where the business process owner became so excited by the discoveries he made in the workshop that he went right out that day and began to institute changes in workflow ahead of deployment of the new application!

d. They do analysis for as many user types and subtypes as is feasible, given the scope limitations, timeline, and resources for the project.

Normal Practices

a. To create an application, all teams have to do some sort of workflow analysis, which is often tasked to the Business Analyst to collect from managers as a part of their process of use case writing. This might take the form of the analyst sitting with various managers simply listing out tasks, or by requesting that business unit managers send them textual lists of core tasks and procedures, etc. But no real analysis is done.

b. Some teams do a deep and detailed analysis using a currently fashionable formal methodology, but the results are so deep, extensive, and detailed that they can't easily be brought into the application design process. Often only one or two people on the team have the responsibility of integrating the information, and the core team is removed from both the process and results.

c. In many teams, users are only involved once in a one-on-one "data dump" of current work and task flows, and they are not involved in the process of improving the flow, nor are they involved in any of the interface design.

d. Due primarily to established company culture, teams are often reticent to evangelize business process or operational changes that arise as a result of work- and task-flow analysis, fearing that it will send a red flag to stakeholders, and that the project would lose its timeline or scope. They tend towards job protection rather than productivity gains or quality of end result.

e. Typical teams who do this analysis do not involve enough user type roles or sub-roles. This tends to create an application that dictates workflow through its generic architecture and interface design rather than facilitate the productivity of diverse users.

4. Creating User Type and Subtype Strategies

See the "Developing User Strategies" section for a full description.

5. Bring It All Together—Design As A Team, Design in Meetings and Workshops, Prioritize

Perhaps one of the greatest challenges development teams face in terms of user experience design isn't too little information, but too much information. How does the team translate a large city phone book's worth of research and requirement information into practical application interface architecture, navigation paradigms, and core work screens for each user type, etc.?

There is great skill and art involved in user experience design. It is legitimate to say that great applications are created by teams that have great individual developers and designers. But there are teams with very innovative and skilled members who don't create great applications. This is often because the team doesn't employ effective group design and decision-making processes.

One of the greatest assets a multiplayer, multi-role, multidiscipline team has is its diversity. Here the sum is definitely greater than the parts. Each role and each player has strengths, weaknesses, and biases that they bring to the table. And each team role also brings a critical perspective to the user experience design process. Great teams have practical ways for the information and perspectives to be shared with other members in a timely and effective manner.

Design as a Team

Many teams start to work more virtually by the time interfaces begin to be developed. Often they have a couple of members create visual screen designs, such as wireframes, and then email them around to each other for response. There is a fundamental problem with this method of design. When almost anyone responds to a design, they go into "critic" mode first, pointing out little details that need fixing, etc. But it is a forest and tree issue—they fix many of the trees without realizing they may be in the wrong forest entirely. The screen paradigm may be wrong, may not be flexible to other user roles, etc., yet individual team members critiquing the design on their own are often not able to appreciate or address the broader user experience paradigm and instead focus on small details.

Also, it takes an incredible amount of linear project time to work in a design-present-respond-reiterate mode. When the entire team, or a core subset of the team, designs together, they can arrive at consensus on many key decisions in a single meeting which might take weeks to achieve when the team responds individually and on their own schedule.

- This is no small thing. In a typical iteration cycle, where designs are sent out to team members for comment, it will take anywhere from two to three working days to get feedback and begin on another iteration. In a team work session, with all eyes on the same whiteboard, it is easily possible to go through ten or more iterations of one screen, or navigation concept, or one user experience convention, in a single session. Do the math.

Design in Meetings and Workshops

Workshops are essentially working meetings, but with a structure. The most important key to successful workshop design meetings is leadership. It is a unique skill to be able to guide a group of co-team members, users, and

business representatives through a successful process where no particular players dominate, where often quiet or shy members with good ideas are heard, and where good design is the shared goal.

Some of the best practices in workshops are:

- Good leadership (a leader who knows how to guide groups to decisions);
- Team size (if the entire team is large, over 10 people, it is more effective to do workshop design with a slightly smaller subset of the team);
- Choosing the right interfaces to design as a group (usually strategically important interfaces for each of the user types or subtypes);
- Striking a balance between under-detailing or over-detailing designs; and,
- Involving actual users in the user role for which the screens are being designed.

Weigh and Prioritize

User research results, user and business requirements, work- and task-flow analysis, desired features and activities, all need to be integrated into application design with the proper emphasis and priority, in order for the application to be highly usable and productive.

Priorities must be arrived at methodically, and factor in input from all key influencers of the application development process—business, users, and IT.

Best Practices

a. The best teams design interfaces as a team, or as a sub-team of core member representatives from each team role. In doing this, they innovate, build consensus, and create buy-in early on, take advantage of their naturally varied perspectives and skills, and save linear time on the project.

b. The best teams recognize that design workshop meetings need to be planned, have skilled leadership, and most importantly, involve actual users from the user roles for which the application is being designed.

c. The best teams put a majority of their time and energy during the experience design phase into creating visual designs of screens for the group design process. There are many techniques they use—most common of which is a wireframe, or a full visual design. These teams sometimes create project war rooms where there can be whiteboards, walls full of sketches, workflow diagrams, wireframes, etc.

d. The best teams methodically integrate recent user work- and task-flow data into the actual design of navigation systems, user role areas in the application, and individual work-area screens. They know that a great application has to work like the users work, with a natural and productive workflow.

e. The best teams have systematic methods and tools for integrating all the various inputs for user interface features and activities into one place. These tools facilitate weighting and prioritization, and are able to summarize the core features for which each interface area or screen is being designed.

- (See Documentation: #5. Translate business requirements and user requirements into features, and then prioritize them.)

Normal Practices

a. Typical teams work virtually after an initial kick-off period of meeting as a group. These teams quickly move into a create document-get feedback-revise design style of working together.

- Because of this, linear time is increased on the project as team members are caught up in continual revision cycles.

> There is also a quality issue as well as a time issue. Enormous time is saved in doing team workshops, because the team can iterate much faster than they can virtually. But another significant advantage to workshops, where team members, users, and business representatives are working together for extended periods of time, is that everyone is able to remain immersed in the information. Even the brightest people will not remember all the key information they should when reviewing wireframes, for example, days apart. They simple can't keep all the requisite facts uppermost—especially when they are working on other things. The continuous immersion of the entire group in all the factors that inform the project results in significantly better designs.

b. One or two team members, who share a particular role or point of view, do the bulk of interface design and strategy.

- Because of this, the interface biases toward the work-and-informational-processing style of that team role. In the case of a strong IT orientation, the navigational paradigm tends to divide the application up into logical "data types" (e,g., customers, work orders, invoices, etc.) rather than into user role workflows (e.g., take customer order, create work order, etc.)

c. Many teams rely solely on written documents until actual screens are prototyped, and create no visual screen designs earlier in the process.

- It has surprised us how often this happens. It seems self-evident that when you are designing something that is primarily a visual experience (the screens a user interacts with) then using a visual format throughout the design process is essential. This is not to say that textual or spreadsheet-type documentation is not also important, but the visual design is paramount.
- Many team members (often business representatives) who are not able to assimilate and visualize textual documents, such as use cases, do not understand the implications of all the data for screen designs until they see the first prototypes. Then these team members often weigh in with revision requests, but this is very late in the game and costly in terms of resources and timeline. This also means these members haven't really had full input into the early design strategies, and hence the core user experience paradigms have not benefited from the perspective of their role.

d. Teams will often engage in some work- and task-flow analysis, but do not use any methodical system to integrate the results of that analysis into navigational system architecture, user type "areas" of the application, and screens. Or these teams have one group do the workflow analysis and another group design the interface, without much interactive communication (other than detailed documents) between the two.

e. Teams rarely have a compact, accessible, practical tool to integrate all the varied sources of potential user features and activities. When they get to actual screen design, the task of going through all the varied documents is overwhelming, and team members doing the interface design tend to revert to the basic bias or perspective they had before any of the research or analysis was done.

6. Developing Interface Components: Navigation Paradigm, Application Shells and Grids, Modules and Module Components, Icons, Action Controls, Wizards, Widgets, Success and Failure Messaging, Help, and Textual Messaging

Developing the building blocks of a flexible and modular interface is key to the success of most web-delivered applications. A highly modular system allows the most flexibility, and allows for static, or on-the-fly, dynamic screen customization for various user types and subtypes.

The skill and experience of individual development team members in web-delivered application interface design plays significantly, and affects interface component development. Also of extreme importance in this phase, is for user experience designers to work closely with back-end coders and those responsible for front-end display code.

Strategizing the Navigation Paradigm

Navigation paradigms (i.e. left column, top tier, expanding tree, dynamic, static, etc.) are perhaps the most critically strategic component of enterprise tools. These applications may have tens of thousands, or at times, even hundreds of thousands of screens in large corporations. Development teams must recognize that there is no perfect system. Their solution will have some inevitable compromises, but the team is able to strategize together for the highest return on their design investment. They constantly benchmark against best-practice navigation system heuristics (is the navigation consistent, zoned, flexible, expandable, customizable, and can it go global, etc.?).

Designing User Type Application Shells and Grids

The key issues with developing application shells and grids is for teams to understand that differing user types more often than not require more and varied work-area grids than the group may initially think (some need multi-column for many small modules, some single column for large spreadsheets, etc).

Designing Modules and Module Components

The key here is again flexibility, especially for unforeseen future needs. If teams work together early on to establish a complete library of modules and components (portlets, tables, scrolling, non-scrolling, forms, text-only, full-width, columnar, minimal, floatable), and prove them out with actual HTML proto types, they save a great deal of wasted time and effort later.

Designing Icons, Action Controls, Wizards, and Widgets, Success and Failure, Help, Textual Messaging, etc.

Teams who early on develop their building block library with icons sets, controls, and decisions about key interactive elements, such as form validation, and success and failure indicators, accelerate their development process.

Early buy-in from team roles such as stakeholders and Business Analysts is important, as these items wil become just some of the myriad revision details requested later. Matters of taste arise ("I think that icon looks like a hand, not an arrow") between team members, and the earlier they are resolved, the better.

Best Practices

a. The best teams take advantage of their skill and experience to develop a solid set of interface building blocks early on in the interface design process. In this way, as varied user type work areas are created, team members can select from an approved set of components, rather than have to initiate another round of "present and revise" each time a new need arises. Also, in this way, matters of differing "taste" can be resolved early, facilitating the project timeline.

- A well-designed and easily usable style guide is the practical outcome of this process. The ideal style guide includes display code so that programmers can not merely reference styles but actually "grab" pieces of code and drop them into their work. Additionally, a good style guide will have styles that have not yet been put into use, but are likely to be needed in the future. An excellent example of this, and the best teams think ahead like this, are fourth, fifth even sixth, and seventh-level navigation paradigms. Inevitably, applications grow in size, but most significantly, in depth of detail. A well-designed modular system can easily scale to handle growth.

b. The best teams prototype and test navigation shells, modules, and components early on. In this way, screen designers can choose from a library of tested and approved building blocks.

c. The best teams recognize the extreme importance of navigation systems. They take the time to plan many years ahead for the evolution of the application, and pressure-test the strategy against worst case scenarios in terms of number of pages, and increased and more varied user types. These teams fight for tight rules around the navigation interface so that it will remain consistent and intuitive.

Normal Practices

a. Typical teams develop interface components one-by-one, as the actual screen they are focusing on requires it. They often create a new module or work-area design that a particular user type requires in the process, and have to go back and modify already designed (and/or tested) shells and grids. This is a time-consuming and team-resource-intensive process.

b. Teams often do not prototype and test components until many screens or even full areas of the application interface have been designed. They go into beta development only to discover there are critical problems with the design, and have to return to the drawing board.

c. Very often, because of timeline and resource considerations, no group takes responsibility for fully strategizing and developing the application navigation matrix. This often leads to cumbersome hybrid fixes and add-ons later (usually adding more and more tiers of sub-navigation) as more and varied user-type areas are added into the mix.

- We have all, alas, experienced navigation systems like this. The application feels like a small house that has had extra rooms, porches, kitchens, etc., added on haphazardly over the years. It looks very odd and finding your way around within it is always an adventure.

7. Content Development

Arguably one of the most critical elements of excellent enterprise applications is their content—be it textual, messaging, help information, online training, documents, graphics, images, spreadsheets, email, and more. User experience designers need to address how content is to be delivered to application users. Just as important, when and how is it gathered into the system?

Enterprise Content Management (ECM) is a huge and equally important issue all on its own, but not the primary focus of this study. The best teams strategize early their content infrastructure, entry, and delivery systems.

(The seemingly inevitable dilemma of content meta-data rears its head during this strategy process. Users need and want to be able to easily and successfully filter and search for relevant content, but to do so, that content needs to have specific, detailed, and prioritized classification meta-data.)

Teams often have to strategize around several modes of data entry—by specialized teams and/or through unique data administration interfaces; data entered/modified by typical users in real time and natural workflows; and data captured/updated by the system behind the scenes.

Best Practices

a. The best teams create a system of dynamically customized, multi-user, role-optimized content delivery. Different user types require different content, in terms of volume, language (some user types may call information by one name, others by a different name), and content classification systems. To achieve these goals, development teams must create a content-creation, editing, entry, and warehousing system which is modular, and as extensively meta-data classified as is possible.

- Large amounts of information are available to users via central databases that provide information not just for one application but for many other applications as well. The data present in these databases come from a variety of sources. Teams need to be aware of how specific data added to these databases through their application will integrate with such databases. However, there are other content challenges that tend to fall outside central collections of data, such as email, information specific to a single user type (e.g. an individual work assessment created by a manager), or for work groups of user types (e.g. such as team reports). We tend to refer to this as "distributed content entry." Content entry from a specific user level (such as a manager) can be invaluable. The best teams design for specific content to be where it is most needed and when it is most needed, but, for distributed input, they must provide the means for adding the content as well.

b. The best applications deliver content to users in small and workflow-directed chunks and then offer those users options for more details. The best teams strategize the content on every screen—do users see what they need to see and can they take the next step for more?

- This is one of the most repeated exercises in experience design. Typical screen size and screen resolution allow for only so much data to be displayed at once. (We often advocate a two-screen display set up just to overcome this limitation.) The challenge is to display the optimum summary-type data that will be of the most benefit to the users and then apply various means to allow the user to see more than summary information with one click. This

can include expanding the information contained in a row of data, displaying more detailed information in a separate module on the same page, opening a pop-up window (rarely recommended), etc. Getting this right for a specific user group creates significant increase in productivity.

c. These teams understand that often the best system of content entry is distributed across many users, entered "in real-time" and in the context of natural workflow by the users most likely to have the best content and the capability to validate it.

d. The best application teams put as much design intention into administration-based content entry interfaces as they do customer/client-facing interfaces.

Normal Practices

a. Typical teams do not spend the time and effort needed to parse content into future-delivered user-role-based chunks and/or into well-meta-data-classified systems. The end result is that typically content can only be delivered in one way and in only one volume.

- This results in users avoiding or not trusting content, because these users know the content will either be irrelevant, unwieldy, or inappropriate to their current workflow.

> We wrote before that you could describe most applications as a means to add, edit, or delete information. In all of these—adding, editing, deleting—finding the right information to manipulate, and quickly, is a universal need of application users. When insufficient experience design has been applied to this problem, users either have to scroll through way too much information, or resort to search systems that are often a worse experience than scrolling and paging.

b. Most often teams do not create distributed content entry systems. It is very simple for them to man date that certain internal users update core content, and no one else. This is a very safe system, in terms of business process and content integrity, but it often results in time-delayed and role-based competitive content.

- Poorly designed or poorly administered distributed-content entry systems can result in data inaccuracies, redundancy, and individualized but eccentric data input, which cannot be easily searched or integrated with other data in the database. Because of this, companies are often reluctant to allow distributed data entry and develop centralized data entry groups where they feel more quality control can be exercised. While this does insure accuracy of data, it typically puts a constriction on the speed desired by the end users (imagine them saying, "Where is that data? I sent it off to the entry guys days ago!"), and the ability to have data localized for small work groups.

c. Typically teams put a great deal of energy into service delivery application design, but a corresponding minimal effort into essential data entry administrative interfaces. The best teams recognize that internal users are all equally employees, their time spent on the computer ends up in terms of company ROI, the data they input in invaluable, and as such, their application interface is equally critical.

8. Designing, Creating and Maintaining the User Experience and User Interface Style Guide

One of the most powerful tools development teams can use to create consistently user-friendly applications is an easily understood and easily referenced style guide. This is often bypassed or developed after the application has been designed because of timeline and personnel resource issues.

Our study revealed that applications can, in fact, be developed more quickly with precise and detailed style guides. More time is spent earlier to develop the guide, but from that point on the design work goes more quickly, with fewer unnecessary revisions and less time spent in group decision-making processes. The guide empowers members of the team to be able to work more independently.

It is important to design the guide in such a way that it is accessible to all the different roles in the team. If it is viewed as being just a back-end coding technical document, Business Analysts will not refer to it—yet they are making daily decisions about the application interface.

Best Practices

a. There is one!

b. Typically, a few strategically chosen screens are created and approved by the team (and these are also prototyped and tested) to begin forming the core of the style guide library of shells, modules, components, user interaction specifications, UI elements, coding and scripting resources.

c. In the best teams, each role from the development team contributes to the guide from their role perspective, but there is at least one owner of the guide. It is approachable by all members of the team, and is most often online for ease of access and updating.

d. The style guide is evangelized by the Team Leader and referred to by everyone on the team religiously. It is a practical tool which serves as a benchmark for design decision-making.

e. As new screens, modules, controls, components, and content are developed or modified, the guide is immediately updated.

f. The style guide isn't just a technical coding and scripting guide. It has extensive user experience styles specific guidelines for how the user interacts with the application, navigation, action controls, form validation failure and success, messaging, and help, etc. It has pixel-perfect, annotated shells, screens, modules, etc., that team members can choose from while flushing out the application screen designs. A great guide also has complete display code components that programming teams can download from the guide for development.

g. In a large application development process, with multiple development teams, each of the teams con form their design process to the style guide.

Normal Practices

a. There simply isn't a User Experience and User Interface style guide during the development process. A style guide is often developed after development has concluded. In that case, it doesn't function as an application creation tool, but as a maintenance tool.

- This happens because it is only after the application begins to go into acceptance— and beta-testing, etc., that members of the team become freed up to put their attention to it.
- Better this than nothing, of course. At least ongoing revisions will have the benefit of the styl guide, which will tend to mitigate a drift in styles over time and as new team members are added.

b. Teams that do develop a style guide, often do so on a screen-by-screen basis as they develop the application, rather than thinking ahead to all the various modules, controls, content, etc., they will need to flesh out a design.

- Specific solutions which are developed should, of course, form the core of the style guide. But in every session where solutions are determined there is an opportunity to ask future development questions such as, "What if there are more links required to go into a module header than we have here? Do we allow the links to wrap and cause the header to double in vertical height? If no, then we need to establish this as the maximum number of links. If yes, then we need to decide if we want to limit the expansion of the header to one additional row or allow more." Taking the time to think out other possibilities saves a lot of time down the line when differing uses of established user conventions inevitably arise.

c. Often the guide is almost exclusively a coder or scripter reference tool, with very little user experience and marginal user interface sections. As such, non-IT development team members might scan through it once and never look at it again.

d. Teams will often not take the time to detail out user experience interactions; this becomes especially problematic in large applications where multiple teams design the user experience slightly differently.

- Describing user experience conventions in a style guide takes more time and care than simply putting user interface components into a style guide. But the extra time and care more than pay for themselves. There are many a large application where the building blocks—the user interface pieces—are all used properly, yet the overall user experience is very confusing and inconsistent.

9. Work Area Interface Development

The core of any web-delivered application is the work areas designed for the various user types.

Creating a Design and Delivery Matrix

The best teams create a document that outlines what work areas will be designed, when, and in what priority order. The more strategically important screens come first and/or the screens that will be challenging for the back-end coding and browser-scripting groups.

Screen-by-Screen Development: Workshops, Whiteboard Sketches, Wireframes

The best teams conduct a thorough, screen-by-screen visual design process. Workshops focus the team, User Reps and Business Analysts on the specific features, functions, modules, and user interactions that will take place in specific screen work areas.

The best teams use whiteboards, flip charts, Post-it notes (re-arranging them together on a wall or white board), and/or PowerPoint prototypes that have been created for group discussion, etc., as tools to visualize work area interfaces.

The workshops rough-in the screens to a high degree of accuracy and consensus, allowing team members to develop detailed visual designs of screens, such as wireframes, for the team to iterate later. The workshop approach saves linear project time (less iterations required after the workshops), improves the quality of the user experience (a great deal of iteration takes place in the workshops), and fosters innovative and creative ideas to come forth.

Integrating the Use Cases into Interface Design

The best applications come from use cases that are developed after user research is done and user requirements specified, after key user strategy decisions are completed, and after key wireframes or screen sketches for each core user type have been designed.

In this way, the core project team maintains control of the user experience, rather than leaving essential interactive elements to be designed by the programming team (and often the programming is out-sourced, often off shore).

If programming teams design interactive elements of the application, it tends to result in a diluted, generic user type strategy, sequential index-based screen flow, and navigational areas based on information type rather than user workflow.

Designing Multi-User Type Work-Area Screens

There are always work areas shared by most or all user types (task-management screens, search-and-search results screens, group communication tools, file logs, reporting, etc.)

Developing the preliminary interface designs for these screens and tools first, or early on in the process, allows for a seamless integration with the more specific user-type work area screens developed next.

Designing Specific User-Type Work Area Screens

When teams focus in on the actual work tools for the various user types, the best applications result when they bring in User Representatives to participate in the design of their area. When designing for a specific user type, the best teams refer constantly to that type's user strategy, to user research results, and to work- and task-flow analysis.

Best Practices

a. The best teams work from an agreed upon design and delivery matrix, and work in a prioritized order. This design development order is often dictated by areas that have longer or more complicated development on the back end.

b. The best teams often work in a system of waterfall development (more will be said of this in the UI development section): UE design, user-testing, revision, pass to UI, design UI, test UI, revise, pass to coding, etc.

c. The best teams have methodologies for the group to visualize screens while they work through fundamental user experience design issues together (whiteboards, wireframes, prototypes, etc.), and don't rely on textual documentation alone.

d. These teams solve the user experience issues and design the work management screens shared by all user types first, so there is seamless integration of these screens with more user type specific work areas.

Normal Practices

a. Typically, teams will pick a primary user type and begin to design a more or less complete set of interfaces for that user. They don't take the time to look at the entire project and create a design order strategy with documentation.

- Because of this, the screens designed for that user type which turn out to be shared screens by many or all users, are oriented specifically to that user type, and will have to be revised and iterated often as the UE process continues.

> It is, of course, important to make the interface work best for the predominant user type. However, if the predominant user type is the only type under consideration throughout all of the design process, the team will miss the opportunity to create slightly varying user experience designs as the process unfolds for other important user types. This leads to the usual result that the other user types, if considered at all, get referred to the next version, because the team has run out of time. If the other user type variations are considered within the flow of the design process, however, the team stands a reasonable chance of getting those alternate experience design paths for other user types implemented.

b. It is typical for use cases to have been developed previous to user experience design sessions, and the use cases tend to be written in such a way that they lock in, or predetermine critical work area screen user experience features. This tends to make the team reticent to go into a process that would require altering the use cases again, so basically all they can do is recommend minimal refinements, basically supporting a user experience design that one or two authors of the use cases have determined.

- If the team is used to rapid iterative design processes, having the use cases written prior to screens being visually designed is not a serious problem. However, we have observed that the use cases take such a major push to get written that most teams will not be able to take the extra time to iterate the use cases as visual screen designs suggest different solutions. Ideally the two are done together, and highly iteratively. However, if one needs to precede the other, we recommend that your team focus on the visual designs first.

c. A group from the team designing user experience screens has often not gone through a prioritization process with the back-end and browser-code developers, and are unaware how critical it is for those teams' members to receive some work area screens sooner rather than later to stay within timeline and scope.

10. Iterating the User Experience Design

As has been discussed in the user requirements-gathering and user strategies sections, perhaps the greatest tool that development teams have for creating great application is extremely simple: design, test with users, reiterate the design.

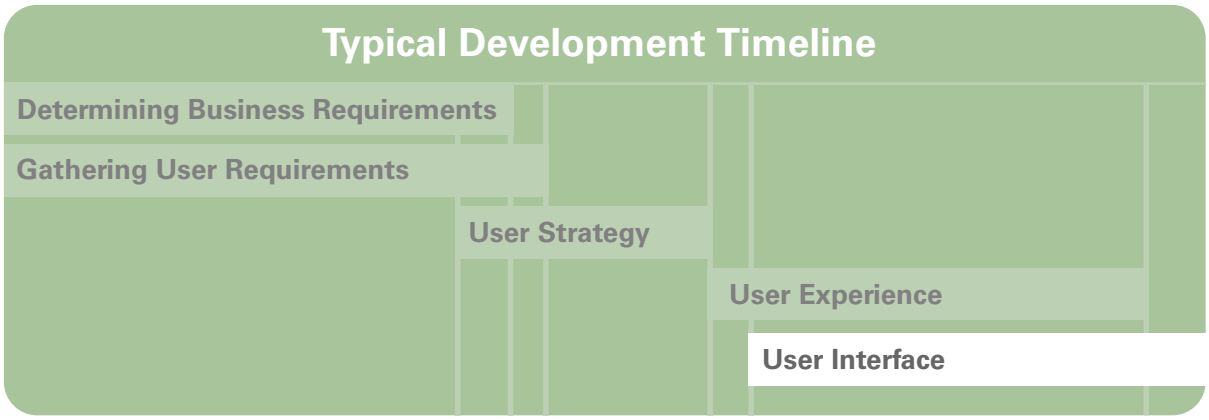
The same holds for user experience design. Simply stated: The best teams show designs to users (UE designs, full UI integrated designs, etc.), whether formally or informally, in small groups or individually. They listen to the feedback and are willing to iterate their designs.

Even if the teams do the initial research and requirements-gathering with typical users, they will get into a timeline crunch by the time they enter UE and UI phases, and will not take the time to test their concepts out. They think that when the application has some functionality in a beta form, then they'll take some users through it. But by then it is too late in the process for this feedback to significantly improve the user experience design because there is too little time left to make the required changes.

- The feedback from users at this stage does not have to be meticulous. The more details your team can go over the better, naturally, but even informal testing with users is invaluable. Establishing a User Representative group makes this very doable. A group of representative users, whose day-to-day work involves the application in development, or the business processes the application will support, and who have been involved from the beginning of the project, can give insightful, critical, and enormously helpful feedback on screen designs-in short timeframes and with minimum formality. If your team does have the time and budget to do more methodical testing all the better, but don't over look the value of quick, informal feedback from knowledgeable users.

Designing the User Interface

The User Interface (UI) is the sum of the actual visual components through which users interact directly with the system. The UI is applied to, or is a more refined expression of User Experience (UE) design strategies—the navigation systems and controls, modules, components, images, and textual content.



1. Team Make-up and Processes

Historically, enterprise development teams have placed minimal importance on UI, focusing most of their efforts on business and functional requirements, and then only minimal focus on UE and interface design as specified in use cases and business requirements documents. The main concerns that teams have had with the UI is download and display speed, browser compatibility, compatibility with back-end data systems, and simplicity in front-end scripting.

- Put simply, user experience design and user interface design are often an afterthought—particularly user interface. If use cases are written, there is at least some attention given to the users' experience. However, the actual end development of the user interface-what the user actually sees-even if there is a UI Style Guide already developed-is left to the programming team to work out. It is considered to be primarily an exercise in coding.

The user interface design process is the caboose of a long train of team processes and decisions in the application design flow. The team players are usually tired, if they aren't 100% resourced to the project, other work is piling up, and/or their timeline has been compromised. They are often under high pressure from project stakeholders and company leaders to “just get it out the door.”

Even teams with highly skilled UI designers involved (often outsourced) can end up with unsatisfactory interface designs. This may be a result of the team as a whole not understanding the high value of user interface design and subsequently not giving this phase of the project the time and energy it requires.

- Often teams, whose Team Leader, or other key team members are current or former programmers, see the importance of the application to be what it can do—not how it allows the user to do it. Teams with a heavy coding perspective are often incredulous when their newly implemented application gets poor

user reviews. They will say things like, *"This is a fantastic application; we were able to do things in this application we were never able to do before."* We in no way want to downplay the importance of great code and innovative back-end solutions. But great code and innovative back-end solutions are only half the equation. User-centered experience design and user interface design also have to be topnotch as well—otherwise users will never be able to really appreciate what the application can do.

UI design is notoriously hard, as is all visual design, for groups to approve easily because it has a high emotional element, and matters of personal taste can creep into seemingly objective decisions about matters of usability and productivity. It is sometimes politically complicated for them to vote for a UI design that a stakeholder or manager doesn't like.

- There is a joke among UI designers. Question: *"Why are eighty percent of internal corporate interfaces 'blue'?* Answer: *"It is the only color that ten members of a development team can vote for without fear of losing their jobs."*

Best Practices

a. The best teams have team members (or this skill is outsourced) who are trained and experienced in the skills of user interface design (including graphic design skills).
    - In these teams those responsible for UI development also participate in (or immerse themselves in the documented results of) user research, know the user types intimately, and are key players in the visual design of screens, such as sketches, wireframes, or full visual design.

b. On the best teams both the Team Leader and the UI Developer(s) help moderate and direct group decisions in matters of taste vs. excellent UI design. Their goal is to achieve the highest possible usability for the various user types, and not to satisfy matters of taste or approval politics by development team members and stakeholders
    - We often recommend that the core team not be the strongest influence on the decision. Put it in front of users and go with what they like—even if the team as a whole favors a different visual design style. This is not about what you or your team likes, but what your users find the easiest to use.

c. UI design elements, styles, and conventions are communicated clearly and in detail to new and/or co-development teams who take part in current and future phases of the application development project.

d. The best teams "stage" decisions with the group so that when the UI is first presented, the group is not simultaneously deciding on core UE issues (navigation paradigm and details, etc). The leaders have set up a process where decisions flow first from a fairly detailed UE, and then on to UI.

e. These teams usually have a "waterfall" development and approval process, especially in larger applications, versus a strictly "linear" process: a group of UE visual screen designs (such as wireframes or sketches) are developed (the volume of which is partially defined by the abilities of the approval team process), then, once approved, that group of screens moves to UI development, then to browser display code validation and scripting, then to programming, etc. After one group shifts to UI, the next set of UE visual screen designs goes into process.

- The teams spread the time-intensive approval process for UE and UI over a more reasonable time period in this way, rather than have an overwhelming number of UI elements that need approval all at once.
- And perhaps even more importantly, when UI design refinements are discovered to be needed in later processes, it is relatively easy to back-fill those details across a smaller set of screens.

f. The best teams work together to prioritize the parallel development approval process so that screens implying more intense and longer, or dependent back-end development, can get through the process earlier.

g. The best teams engage in an "iterative" UI process that includes input from User Representatives or user groups. These teams know that inevitably the users will have good input, and they are ready to iterate the design again to get it right.

Normal Practices

a. Typically teams have no UI Developer trained in graphic design and they only have one IT developer on the core team who has skills in UI development tools (such as PhotoShop), who puts together a set of html templates and minimal graphic elements to form the base library of UI elements to be used in the application.

b. Sometimes, teams will have a graphic designer from the marketing side help work on the UI (or they will outsource a graphic designer for a very short time). This person usually has limited or no serious training or experience in developing graphic elements for browser display. And/or they have a limited knowledge of the very unique visual best practices for long-term use applications vs. traditional Web sites. And/or they tend to design the internal applications to styles that have been specified for the company's consumer-facing Web sites.

- When graphic design is outsourced it is imperative that the designer be highly experienced in web-delivered application design. It is typically an exercise in minimalism. Experienced graphic designers know how to use the minimum of color and shape for the maximum in clarity, ease of use, and ergonomic design. Users often use the applications many hours a day. The best graphic design elements provide clarity, enhance important information and important controls, without standing out. Inexperienced designers tend to want to make too much of a visual splash, which often takes up too much space, and over time becomes a strain on the users' eyes.

c. In typical teams the UI Developer(s) are brought in after the use cases, wireframes, and sketches, etc., are fully specified. Not having participated in these important processes, or in the direct user research (if any has been conducted), they are removed from the full wealth of gathered user and user experience information and cannot really integrate it into the subtleties of their designs.

d. Decisions on UI elements are ruled by the "taste" preferences of the most vocal or dominant players on the team, and the resulting approved designs can only reflect those players' relative sophistication in terms of UI best practices.

e. Teams withhold visual designs from key players for too long, resulting in many revisions of the visual design. This often results in a lowest common denominator decision being made to speed things up. In large companies, key design decisions and details are not propagated early and across multiple teams, resulting in inconsistent applications having subtle but important interface variations across various areas.

- It is an unfortunate fact that stakeholder-level, or upper management-level decision-makers will sometimes have a lot of interest in what the application looks like—to the point where CEOs have been known to overrule the team's design decisions—which have taken weeks to arrive at—in just a few minutes. It is better to float user interface visual design out to all key decision-makers early rather than late. Better to surface strong opinions when your team can respond to them, than to find out in beta.

f. Many teams first present, for group approval, several screens in full UI (in color, full-sized, including all the proposed graphic elements) when most UE details have not yet been approved in wireframe form. Often they have seen "snippets" of wireframes in Use Cases or other documents (knowing basics of navigation paradigms, control elements, etc.), but when they see it in "living color" they are over-whelmed with the complex set of approvals that need to be made.

- Sequential decision-making—user strategy to user experience to user interface—makes the whole process go much more smoothly. We have witnessed many a meeting where crucial experience design issues need to be decided, but the entire meeting is sidetracked by shape, color, or icon design issues because the user interface is being shown at the same time as the user experience. Once the user interface components are proven out, remaining experience design screens can be presented in full Photoshop mock ups, but prior to the user interface being settled, we have found black and white diagrammatic wireframes to be the best way to present experience design.

g. Discussions at a screen review meeting will often range from UI topics like "I don't like that shade of blue" to UE topics "I thought we were going to have six major areas in the left-column navigation, not the eight I'm seeing here!" In these teams, leaders are not skilled at setting up "staged" decision processes for the group, especially for the more emotionally packed UI approvals.

h. User interface development is often jammed into the remaining time the team has in its schedule.

- Whether or not the team has a philosophy of iterative design, and/or one of testing UI with User Representatives, at this point it isn't an option, and the thought is that users can comment in acceptance-testing. Again, by then, there really isn't any option for serious iteration or back-filling of UI design refinements.

2. Graphic Design and Information Hierarchy

Development teams rarely fully understand the importance and subtlety of visual information presentation hierarchy, and tend not to have players involved with the UI who are experienced in the arts of graphic design layout and typography.

- Studies have shown that even just the design of subheads in a body of text (font size, type, color, etc.) and their relative spacing on the screen can increase users' capability of quickly scanning and locating pertinent information by up to 15%. Multiply that figure by 5,000 all-day users in a typical corporate call-center environment and it is easy to understand the kind of productivity gains that can be made from UI refinements.

Best Practices

a. The best teams give qualified UI designers room to refine and improve sketch and wireframe concepts with visual elements, knowing that often a well-designed UI can simplify the look of screens, increase their long-term ease of use, help define informational hierarchy, and even allow for higher data density than was conceived possible in UE sketches and wireframes.

b. On the best teams, all players participating in creating and/or approving UI design know the importance for a flexible, expandable, customizable, modular system of UI elements.

Normal Practices

a. Sometimes teams have qualified UI designers, but are given such a tightly specified user experience design that they have no room for development. All the specialist can do is fill in with color, and/or design some icons, etc.

b. Teams will at times approve of designs early on that satisfy a few screens, without strategically planning ahead for expandability, customization for other user types, and other workflow situations, flexibility, and modularity.

- This creates a process of situational graphic design. Graphic designers provide a steady stream of one-off solutions. The end result is a very inconsistent user experience and an almost-impossible-to-create style guide.

3. UI and Display Code

Another important challenge for teams is the all important and complicated interrelationship between UE, UI, and display code. Decisions made by the team on supported browsers, on screen resolutions, on the use or non-use of DHTML, javascript, Java applets, etc., will determine the options for UE and UI solutions that can be implemented.

- This sounds relatively simple, and it is simple to determine a display code specification, but the con sequences of not doing this are immensely complicated. Some of the most complex display code bases we have encountered are the result of late decisions to support alternative browser types or earlier browser versions than were originally assumed. Creating display code to behave the same way across multiple browser types and browser versions is extremely difficult. Adding one browser type that

varies from the mainstream (IE and Navigator) sometimes requires entirely different display code—not just cascading style sheet (CSS) variations.

> Late decisions to support other browser types, or browser versions, can also require a complete revision of the experience design. Many experience design conventions, of which a team will want to take advantage, can only be supported by more recent browser versions. Once a team believes they know what display code capabilities they have to work with, they can make key decisions about experience design conventions. Reversing that decision in mid-development can rip the heart out of a carefully constructed user experience.

Best Practices

a. On the best teams the UI Developer(s) know browser compatibility issues and the front- and back-end technical requirements and limitations established for the project—and/or they interact intimately with UI Developer(s) who are front-end display code specialists.

- Someone, if not the UI Developer(s), has to be able to create excellent display code to prove out the screen designs coming out of the experience design process. Once a certain number of screens have been validated as able to be display-coded and able to be implemented into the back end, then experience and interface design can proceed without validating every screen.

b. These teams do not go into full approval process on UI designs until key screen behaviors are proto typed, tested, and validated, in terms of both back-end and front-end display code benchmarks.

Normal Practices

a. When UI Developers are brought into a project well down the line, or they come over from the marketing side, or are outsourced, they are often minimally aware of complex browser compatibility issues, loading and display speed issues, and established front-end code technical requirements. This results in designs going through multiple time-wasting revisions as other team members detail through the display code problems with the designs presented.

- When this happens it can let all the air out of the process. It is very hard to remain creative and engaged in experience design and interface design phases, if the team has to comb back through everything that has been done, and then eliminate—and come up with new solutions for—user conventions or interface components that cannot be implemented. If this happens more than once or twice teams tend to get demoralized.

b. Typically, teams will go through a full review, reiterate, and approve process on a UI that has not been prototyped and validated for display code. When display code validation finally happens, later on, changes are inevitably made and the screen has to go back into the approval process (or worst case, it just goes out with the mandated changes and often other teams members aren't even aware of it until they see it in beta-testing).

- By this time in the process schedules are very tight—or the team is already over schedule. It simply isn't realistic to reapply the design process to the affected screens. Triage methods are applied instead, and the end result is often missing many of the best user experience conventions.

4. UI Development Environment

Many times teams are developing in an environment that has its own library of visual components, such as J2EE. These environments, when used well, can create consistent and usable interfaces (when applied to well-designed and detailed UE interface strategies). Their limitation comes in that these environments are seriously lacking capabilities in the subtleties of UE and UI design-data density variations; typographic variations; use of color, tone, and hue variations; use of graphic elements, such as keylines, bullets, icons, module shape and size variation; font size, color, and type variations; etc.—to create excellent informational hierarchy and long-use refinements. It is a one-size-fits-all solution which doesn't automatically create the ideal, usable, or most productive environment for all user types.

Best Practices

a. When developing in an environment with an established visual library, the best teams consider that these are visual elements and tools, and that their existence does not remove the need to do serious and detailed UE design before applying the tools.

b. These teams often consider making customized elements to augment the existing library when and if required for good usability.

Normal Practices

a. Typical teams developing in an environment with an established visual library may actually use the existing sets of visual elements and/or examples of screens created by those elements as guidelines for designing the UE. They may copy and refine example screens provided by the environment provider.

- Teams often assume, because this library of tested and established visual elements exists, that there isn't a need for a lot of UE design strategy and detailing to be done.

5. Corporate Branding and UI Style Guides

Development teams face a lot of pressure from the business side, from marketing, to have their UI designs adhere to certain standards and styles—but these standards have often been created by groups not fully skilled in laying out usable and productive internal or B2B application interfaces.

Best Practices

a. On the best teams, the UI Developer(s) and the team balance on a fine line of integrating or resonating with general corporate identity (CI) guidelines, and customer-facing online branding requirements, and the usability issues surrounding longer-use internal applications. They recognize the importance of a unified, professional look and feel across a corporation's customer-facing and internal screens, and most importantly, the importance of a strong and well-defined corporate culture.

- The best teams are willing to challenge existing standards and fight to improve and evolve them when they do not facilitate good application usability and productivity.
  - > Many corporate style guides do not include application-specific guidelines. They typically address printed materials, the public website, and company intranet. The

best teams work with the corporate guidelines group to establish application-specific style guidelines that can subsequently be applied to other application development projects. Because of the frequency of use (users are looking at them for hours a day), and the typically much higher information density and complexity of web-delivered applications, web-delivered application guidelines should be different from (but clearly resonate with) other corporate guidelines.

b. Whether creating from scratch, or revising an application, the best teams create/modify a concise, available, practical, and consistent UI style guide early on, and in parallel with the development of approved UI design element details.

- In these teams, all subsequent application UI areas—navigation shells, screens, modules, controls, icons, font styles and colors, action controls, widgets, user type or division banners, etc.—are designed and approved strictly to the style guide. New or modified UI elements are immediately added to the guide.

c. The best teams decide early on on adherence/non-adherence and develop to government accessibility regulations, ergonomics, visual personalization capabilities, globalization and localization issues, document these strategic decisions in the style guide, and design strictly to them.

- It is absolutely essential that these parameters be discussed and determined before design begins. While some of these issues can be simple to design for (e.g. allowing people to change type size) some of them (e.g. allowing the application to be localized into multiple languages) require significant awareness throughout the process of design.

d. These teams know that true globalization is much more than delivering the application dynamically in various languages. It includes global area color significance and preferences, iconic symbolism, cultural design styles and emotions, etc., and that these are very significant issues, and that the design, modular layout, back-end systems and presentation layer scripting must support this.

Normal Practices

a. Typical development teams, that do not often include highly skilled UI designers, do not have any concept of the importance of having their internal application tools resonate in terms of the visual design with corporate branding and CI. They have no sense of the importance of "one company, one experience," or of the emotional and ROI importance of a strong and unified corporate culture (which good and consistent visual design strongly helps to create).

- Their basic sense is "If the tool works, well, fine. What does it really matter what color it is?"
  > Nothing illustrates this better than a tour through the applications in use inside a typical company. In most large companies, with hundreds of applications, you will be hard-pressed to find two applications that have the same style guide. The savings in training time and costs alone should make companies work hard to create highly flexible, yet in-depth application style guides.

b. Most often internal development teams take it as a fact of life that they will simply adhere to any existing internal site styles and standards, don't really care whether these standards were developed with web-delivered applications in mind, consider it is just part of the game of developing within a corporate environment, and wouldn't dream of challenging these standards for fear of putting their jobs in jeopardy.

c. Also, when there are existing internal Web standards in typical companies, they have often been developed in the portal days of early internal sites, and are not really effective standards for more complex web-delivered applications.

d. In large companies, there are often existing requirements to include on every internal screen—whether portal or application—one, two, three or more division banners and navigation bars at the top of the screen.

- Typically, internal teams can't even begin their users' work areas any higher up than halfway down the screen!

e. As stated before, many teams do not create a detailed style guide that includes specific UI development benchmarks, or if they do, it comes at the end as a directional tool for future application development; it is not developed or used as a current project UI design benchmark tool.

f. Often teams do not develop early and specific strategies for accessibility guideline adherence and ergonomic issues (can users select larger and smaller text views, do these optional views break fundamental designs such as table columns, etc). Also, most teams consider visual personalization capability of being at the bottom of a list of potential user options (in terms of value to the overall project).

g. Typical teams are aware of localization needs, but consider this to be mainly an issue of dynamically delivering various text sets out of databases. They may know such basic and important details as that foreign language textual links and content blocks may be up to 30% longer than English ones, that all text needs to be dynamically delivered from databases in text format rather than graphics, meaning even action buttons must be able to "stretch" and accommodate this text—but they do not consider important elements such as regional color preferences, iconic styles, and local cultural design preferences.

Documentation

Documentation is crucial to the application development process. A significant number of books have been written on the subject and numerous applications have been developed to support the documentation process. In the end, we did not try to rank the effectiveness of specific documentation formats, nor did we rank or identify specific support applications, because what we observed in our study showed that the documentation formats adopted, and the support applications used, varied primarily because of the scope of the application and the combined experience of the application development teams we studied—and we did not see a clear correlation between the use of specific documentation formats or support applications, and a resulting superior user strategy, user experience, or user interface.

However, what we did identify were best practices for documentation that cut across all documentation formats and support applications. The best practices we identified were more about method than format. Teams that followed many or most of the best practices we identified, and had superior applications as a result, used a wide range of formats and support applications.

Best Practices for Documentation

1. Keep documents as simple as possible.

a. The best teams had precise, minimal, and accurate documents. The larger the application process, the more documentation it generates. It is important to remember that documentation is not an end in itself. Its only purpose is to communicate key information to the developers of the application so that information can be incorporated into the application.

- Keep additional documents (which support the main or top level documents) "out of the way" to avoid confusion. Make sure they are accessible, but clearly identify them as support documents to main documents.

2. Share documents in group meetings

a. Documents alone do not communicate a holistic vision of the application development project to all team members. Any key document should be verbally and thoroughly shared with the entire team, allowing time for questioning and explanation. Any items, which are vague, incomplete or in dispute, are clarified or resolved, in order to avoid the same items coming up again (and again) in succeeding phases of the development process.

- The best teams recognize that although this is a time-consuming process, it is also the only effective way of catching all issues and resolving them before they become significant problems. It is also a primary way in which teams develop a holistic understanding of the project.

3. Communicate priorities of importance as well as appropriate detail in all documents.

a. The best teams do not lose focus on the highest priorities of the application amidst the avalanche of detail that inevitably flows over them during the development process. The best teams structured their documents so as to be able to indicate priorities.

- Priorities may be indicated by a ranking scale such as 1 to 5, etc. (See #4 below)
  - > When features are ranked by their importance to the user, as well as their importance to business, the user experience design will result in greater usability and productivity.
    - In simple terms, the features which have the most importance to users or business should have the most prominence on the screen and those with the least importance should have the least prominence. If not prioritized, all features will take on equal importance.

4. Determine and document display code specifications at the beginning of the development process.

a. The best teams resolve any questions about what browser versions, operating systems, download speeds, and screen resolution the application must support (including strategies for supporting multiple layers of display code support made possible by "sniffing" user settings), as well as the dis play code types ((X)HTML, DHTML, javascript, Java applets, etc., that can be used before user strategy is even considered. In the world of web-delivered applications, user strategy, and especially user experience and user interface design, are massively influenced by these factors.

- Once the display code specifications are determined, a good team can work within any combination of these factors. However, good design can be derailed if these factors change during the process, or are left undetermined.

5. Translate business requirements and user requirements into features, and then prioritize them.

a. The best teams develop a features specification, which flows from the business requirements document(s) and the user requirements document(s).

- Business requirements are most often stated as high level end goals than as specific features. For example, a business goal may be to personalize an application by identifying the user by name. A feature specification describes this more specifically, such as:
  - > User's name will appear on every page.
  - > User's title, when available, will appear with user's name.
- User requirements are most often stated as wants, needs, or wishes. For example, a user requirement might be that they want more ability to reuse a report they'd previously created. A feature specification describes this more specifically, such as:
  - > User can "Save Report as Template."
  - > User can "Select from List of Templates."
  - > User can "Rename Template."

b. The best teams prioritize the features based on rankings from three sources: business, users, and IT.

The input is typically in the form of numeric ranking; in order of importance from business and users, and in order of doablility from IT.

- Combining or averaging input from all three ranking sources is a very effective way of determining the optimum feature mix for the application.
- Giving users a place in the prioritization process insures that the features important to users make it into the application. The end users themselves do not rank the features but members of the team who are closest to the users (such as the BA(s) or UI Developer(s)) provide input from their perspective.

6. Start as soon as possible developing visually-oriented documents, such as sketches and wireframes, etc.

a. The best teams start sketching and wireframing screens as soon as display code, features specifications, and work and task flow have been determined.

- The final deliverable is an interactive visual interface. Code supports the interactive visual experience, not the other way around.

> Architects first show you what a building can look like, based on a top-level under standing of the purposes of the building, then refine the visual design as engineering challenges and detailed uses of the building are determined, refined, and resolved. Architects do not first settle every engineering and usage issue and then, and only then, show the end design.

b. The best teams use sketches and wireframes as their primary way to explore and discuss the application. Programming Lead(s), Business Analysts and User Representatives begin commenting on and critiquing actual interface designs very early on in the process. The best teams visualize the application—not simply detail it textually.

c. The best teams engage in rapid iteration of visual documents. Wireframes or sketches bring out the need for very specific issues to be decided upon by the Programming Lead(s) and Business Analysts. The sooner these issues surface the better.

7. Create a visually dominated design specification.*

a. The best teams develop specific screens through an iterative wireframe development process. Final wireframes are then annotated and form the core of the design specification. Process steps are shown by creating step-by-step variations of the final wireframes. Required data is shown in accompanying tables. The annotations are sometimes from use cases, sometimes simply descriptive.

- The iterative wireframe process is informed by the features specification, data tables, work- and task-flow workshops and documents, such as use cases or flow diagrams.

*The design specification has other names, and sometimes what we refer to as the technical specification is also called the design specification. As we use the terms, the design specification defines what each screen looks like, how it behaves, and what data can be seen there. The technical specification flows from the design specification and includes significantly more information relevant to middleware and database development and generally breaks the design specification into a larger number of discrete elements.

8. Begin creating (or modifying) a User Experience and User Interface (UE/UI) Style Guide as soon as the global visual design and global shell design are completed or updated.

a. The best teams use the UE/UI Style Guide to keep them clear on already established user experience conventions and user interface components.

- Large multi-team application development processes can't live without a style guide. Significant variations on conventions and styles can easily develop from team to team.

9. Critical best practice documents for developing optimum user strategy, user experience, and user interface design are:

a. Requirements Gathering

- Business Requirements
- User Requirements
- Display Code Specification

b. Feature Specification and Prioritization

c. User Strategy

- Strategy Document
- Global Shell
- Global Visual Design

d. User Experience

- Design Specification
- Wireframes
- Annotations
- Tables
- UE/UI Style Guide

e. User Interface

- Visual Design Elements
- UI Standards (part of UE/UI Style Guide)

f. Technical Specification

Normal Practices

1. Documents become too large for easy assimilation.

a. Team members, confronted with very large documents, especially when they are within another team member's area of expertise, tend to skim the documents once, and then rely on access to the author of the document to get their real questions answered, thus reducing significantly the value of the documentation.

- Something we have frequently encountered is what we jokingly refer to as the "five-thou sand-line help desk complaint spreadsheet." There is valuable user information to be found in help desk logs, but someone needs to distill it down—categorize, give percentages, classify by types of complaints, or break it down into sections of the application the complaints refer to—then it becomes meaningful user information that everyone on the team can absorb—not just the one or two people who nearly went blind reading all 5,000 lines.

2. Documents are completed by the responsible team member, uploaded, other team members are notified by email, and are on their own to read and assimilate the documents.

a. Frequently, as documents are referenced in group meetings, unless the author is present, no other team member is confident of the meaning and purpose of many of the documented items, bringing the group meeting to a halt during key application development processes.

- If the author is the only person who can reliably interpret his or her documents, then the value of documenting information for other team members is greatly diminished.

3. Documents offer enormous detail but the "structure" of what is being documented is lost.

a. Teams end up not being able to see the forest for the trees. Priorities, hierarchies of importance, relationships to key strategies or solutions, are not preserved or referenced in the documentation. The further the document is along the chain of documentation, the less core strategy is conveyed. Sometimes design specifications, and particularly technical specifications (see above) become undifferentiated masses of detailed specifications needing to be included in the application. User priorities and business priorities are included, but end up with no special prominence or higher level of usability and productivity.

- Maintaining a continuing awareness of all the important details—which is to say all the details—of an application development project is beyond most people. Good Team Leaders will sometimes display an uncanny ability to remember enormous amounts of project information, but realistically, most team members begin to enter a blurry world of uncertainty after a while, and really don't remember key decisions, especially important features, etc. When the documents created by the team convey no sense of importance, either, then the team can lose focus down the line.

4. Display code questions don't get fully addressed and resolved until the programming phase.

a. Key user experience conventions, which are dependent on specific display code capabilities, often have to be redone toward the end of the application development process due to display code limitations that were undetermined at the beginning of the process. While fundamental functionality is preserved, well-planned user experience conventions suffer when subjected to late, but necessary revisions.

5. Business requirements are translated directly into data objects, business objects, API modifications, etc.

a. We often found a significant disconnection between the front-end planning process and the back-end development and implementation process. We often observed that the back-end team is primarily interested in the determination of specific business requirements and not the user strategy, or user experience planning. Once they have the business requirements, they move directly to the process of breaking the requirements into the data they will be required to deliver, and begin to plan how that will all fit into existing middleware, or into new middleware development. The disconnection takes place when the front-end planning process is completed and handed off to the back-end developers (programmers and database analysts). At this point, the middleware development process is often so far along that the user strategy and user experience planning can only be used as a general guideline-not an exact specification. We often hear comments such as, "We'll do what we can," clearly indicating that the programming team was never expecting that conformance to specific user experience and user interface design would be required.

6. A thorough wireframing process is rare. Many teams do not develop any visualization of the user interface in detail (or at all), and rely on the programming team to develop the specifics of the user interface during middleware development.

a. This is perhaps the biggest and most consistent failing we found among teams that resulted in poor user strategy, user experience, and user interface design. We were surprised how many teams did not see any interface design examples until the applications were in alpha or even beta versions.

b. Design specs—if they exist—are often just use cases, sometimes with work- and task-flow charts. Many teams go right to the technical specification and skip the design specification altogether.

7. The UE/UI Style Guide is created (if it is created at all) after the application has been completed and serves only to guide incremental changes for the future.

Tips on Outsourcing

Common Outsourcing Practices

You will find numerous references to outsourcing throughout the descriptions of the best practices we identified. However, given the high interest in how to make outsourcing work well (due to the increasing frequency with which companies choose outsourcing in application development) we thought it would be useful to devote a separate section to the subject.

The most common practice we have seen is to outsource middleware development. Sometimes we have seen the middleware and the display layer (the user interface) outsourced. Less frequently, companies outsource the entire project, and provide only the business goals. There are many other variations, but these are the three most common uses of outsourcing in application development we have encountered.

Outsourcing the entire project is by far the riskiest and the least recommended. You and your company are placing too much responsibility on people who are too far removed from your business and your users to be effective. Even if the outsourced team is domestic (as opposed to, say, an Indian, Russian, or Chinese team) they will not have an immediate enough feel for the purposes and the people for which the application is being developed. If they are off-shored, the problem is only compounded.

Outsourcing the middleware and display layer, where the outsourced team's output is delivered to an internal team ultimately responsible for implementation is also risky for the same reasons mentioned above, but primarily because outsourced teams that claim expertise in both middleware development and development of the display layer (user interface) rarely have real expertise in user experience design. They may have competent people who can create display code to work with the middleware they are developing, but the actual user experience will more often than not come out confusing and inconsistent. (We have seen really awful user experiences created this way and are often brought in at this point to try to fix the problems.)

The optimum use of outsourcing that we have observed is to outsource only the middleware to a strong coding-oriented team and, if needed, (due to not having the expertise in-house), outsource the user experience and user interface design to a specialized company. Companies that specialize in user experience and user interface design (full disclosure: Tristream is such a company) have methodologies and practices devoted to user-centered design, usability, and other tools that enhance and inform the application.

If your company has a good core team that can develop the user experience and user interface for an application, working with an outsourced middleware development team can be both successful and cost-effective.

Keys to Working with Outsourced Teams

There are two keys to successfully integrating outsourced teams, especially off-shored teams, into your application development project: involvement and communication.

Involvement

Although at first it seems counter to the whole purpose of using outsourced (or off-shored) middleware development teams, having as many members of the outsourced team on your premises as possible, and involved in as many of the user strategy, user experience, and user interface design processes as possible, will substantially increase the likelihood that your team will work well with the outsourced team.

The additional costs to your company of insisting on outsourced team members being onsite will be more than made up for by better results and smoother operations.

- Until outsourced team members, especially the team leads, have time to really absorb the business goals, user needs, and appreciate the user strategies and user experience conventions your core team has planned, they have no feel for what they need to do to create the middleware or code for the dis play layer. Applications are the result of thousands of small decisions. Teams working in isolation, often in another culture, make countless decisions that affect the user experience. Without significant onsite involvement, by at least their Team Leaders, the team will make those countless decisions without any internal reference to actual people and actual business goals. We have seen an off-shored team do over 50 revisions of a section of an application and be no closer to success on the fiftieth try. Their decisions as to how the user would like to use the application are just projections of their own preferences, and, worse yet, they have such a peculiar view of the application borne of building it, line-by-line of code, that they have no semblance of a typical user's point of view.

Get it in Writing

Outsourcing companies, especially those with off-shored resources, will often tell you that they have significant onsite involvement. What is meant by that can become a bone of contention later in the project. Get a specific agreement as to who will be onsite and how often they will be onsite. Insist on an onsite presence by the actual team leads who will be working on your middleware and specify it in your contract.

Many off-shored companies have in-country representatives that provide account management time to their clients but do not actually perform the work, and who service a number of clients. Communicating work goals to them is worse than communicating work goals directly to the team leads although thousands of miles away, because at least you will communicate fully and without misunderstandings, as you might through a non-involved account manager.

Spend Time with the Outsource Team

It is an excellent practice that greatly facilitates understanding and communication if you send key members of your company's team to the outsourcing company's premises—especially if off-shored. You will get a greater cohesion and better communication if your team gets to know their team directly. Again, this recommendation would appear to be contrary to the spirit of cost-saving that outsourcing is inspired by, but in the long run your company will get much better results—and save money in the long run by eliminating many problems.

Communication

Even if you are able to achieve the greater involvement recommended above, you will still be faced with the need to communicate much more fully than you would with an in-house development team that is located down the hall.

First, your in-house development team will probably have worked together on many application projects and will already have an established expectation for how your team works together—who takes responsibility for what, how the process unfolds, where the team has strengths and weaknesses. An outsourced team will have no such knowledge.

Second, your in-house team has an awareness of the users and the business goals of the application that has accumulated over time simply by being a part of the company. An outsourced team will have no such knowledge.

Because of the implicit knowledge available to an in-house team, communication can be a little looser than with an outsourced team. It is safe to assume certain things, to pass on design specifications with small gaps or unfinished sections, because your team can be confident in one another that these gaps will be filled in intelligently, the decisions made informed by appropriate knowledge.

You cannot assume anything with an outsourced team.

Provide Highly Specific Design Specifications

Since our expertise is in user experience and user interface design, we can only make specific recommendations regarding how to communicate the user experience and user interface to outsourced companies. However, the principle would hold true for any other aspect of the application (databases, infrastructure, etc.) that would inform the work the outsourcing group is doing for your project.

Provide Complete Photoshop Files of All Screens

- Great user experience design pays attention to the tiniest of details. It is not enough to provide user interface components, such as headers, navigation elements, modules, etc., and expect the outsourcing team to assemble them according to textual descriptions only (such as use cases). The best practice is to provide the outsourcing team with exact screen designs accompanied by textual annotations, data tables, and occasionally flow charts.

Provide a User Experience, User Interface, and Display Code Style Guide

- Even with Photoshop screen designs, developers can still render the screen design inadequately unless provided with the user interface elements, exact display code, and descriptions of user experience behaviors.

Optional: Provide a Functioning (Display Code Only) Prototype

- This is not always a good use of time. If the development environment is highly dynamic, the programmers are not going to be able to use the prototype as is, and will have to parse it into many chunks of code anyway. It can be a very useful and informative to everyone, both in-house and outsourced, but it requires significant time to code.

Insist on Regular and Documented Communication

- Do not let the outsourced team "go away for awhile and bring back a finished product." Insist on a methodical review process that allows you and your team to catch problems and make adjustments. No matter how well you follow our recommendations to provide very exact design specifications, there will be holes, there will be things that do not work when the middleware starts to interact with the display code, etc., you want to stay closely involved and insist on seeing output on a regular basis. This will save you an enormous amount of time in the long run.
- Visit the outsourced team's premises often during the implementation phase. Being there will help you catch many things more quickly and surely than relying on conference calls and shared files.
- Be prepared for lots of conference calls. Instead of team meetings on your premises, which require many hours to go over the work in progress, your in-house team is going to spend as much time, or more, on conference calls with the outsourced team. Make sure your team has a projector and that both teams have the up-to-date files under discussion.
  - > Shared screen applications that allow both teams to see whiteboards, or document mark ups, in real time, sometimes work—but we frequently find that they slow down the pace of the meeting.
- Have a clear leader and a clear agenda for every conference call.
- Follow a delivery matrix. Your team and the outsourced team should create a regularly updated delivery schedule of specific screens and sections of the application.
- Create a bug tracker. Your teams should share a bug-tracking application that is maintained religiously. There are no casual conversations across the aisle or over the cube wall that can keep the team informed of progress. You must track it meticulously.

Here to Stay

Outsourcing is here to stay. The economic benefits are simply too great for companies to ignore. We've seen disasters, but we've also seen very smooth collaborations, even with companies halfway around the world. Insist on as much on-site involvement—in both directions—as you can get, and communicate down to the last pixel, and you'll find that many outsourced and off-shored teams can really rock.

Checklist and Best Practice Priorities

Teams often ask us where to begin. No development team we studied integrates all of these best practices. Most teams we encounter in our professional practice have integrated only a handful of these practices.

This checklist, in addition to being a quick overview of the best practices, is ranked according to priority to make your choices of what to add to your team's practices easier to make. This wasn't easy to do. Our difficulty in assigning priority to these practices is that all of the best practices identified are, almost by definition, high priority.

Remember, these practices were the cream that rose to the top, the practices that most contributed to highly usable and highly productive application development. Most of the practices we observed did not even make it into this study. These practices are therefore already the most important ones for you and your team to adopt.

However, we are well aware that a team can't adopt all of these practices at once and choices need to be made. We therefore have assigned the following priority rankings:

Critical
High
Medium

Critical

These practices are absolutely essential. Assuming for the moment that your team did not practice any of the best practices assigned critical, if your team added them all, you would see your usability and productivity take a quantum leap forward in quality. Teams which integrate all of the critical best practices would be among the very best. You simply can't do well without these practices in place.

High

These practices, while not as essential as the critical practices, all contribute to excellence. Teams which integrate both critical and high importance best practices would be the very best. You can do better with these practices in place.

Medium

Many of these practices have a longer rhythm and provide benefits over a longer period of time. They insure good communication, good handoffs to new team members, and continuity over time. Teams that integrate all of the best practices, including these, will not only have success with specific applications, but will tend to foster a long-term culture of success.

Best Practices for Team Dynamics

1. Holistic Understanding—**Critical**

As much as possible, all team members are aware of the entire development process, participate in the entire process, such as determining business requirements, gathering user needs (especially), they participate in the various stages of review (such as use cases, wireframes, design specifications, etc.), and most importantly, participate in all stages of user strategy and user experience development.

2. Fluid and Fast Communications and Decision-Making—**Critical**

The best teams achieve a culture of fluid and fast communications. Application development is highly iterative. Teams that communicate and make decisions quickly are able to process more iterations than those teams bound by more rigid communication processes, and as a result they can achieve greater improvement by processing more iterations.

3. Finding the Value Center: Optimization of Business Goals, User Requirements and IT Capabilities—**High**

The highest ROI for an application requires the optimum balance of these three major factors; underemphasize or overemphasize any one of them, and the application does not deliver its maximum potential value to the organization. Therefore team composition and processes must reflect a balance of these three factors.

Best Practices for Core Team Roles

1. Stakeholder Champion(s)

- Champions the users—wants them to have a tool that "really" works for them—**Critical**
- Views the process as an enormous opportunity to increase productivity—**Critical**
- Champions the application to management—gets funding and resources—**High**
- Deeply involved in the development process—**High**
- Doesn't "hide" when matters get technical—**High**

2. User Representatives

- Maintain one consistent feedback group for planning and initial development—**Critical**
- User Representatives, to be most effective, must be significantly involved in the process—**Critical**
- New User Representatives, asked to test prototypes, also need significant exposure time to the prototype to give meaningful feedback—**High**

3. Programming Lead(s)

- Involved throughout the user strategy and user experience development process (especially if out sourced)—**Critical**
- A key decision-maker: such as system designer, system architect, or lead—**High**
- Comfortable with a "top down," iterative approach—**High**
- Involved in creation of design specification—**Medium**

4. Business Analyst(s)

- Knows the user's needs, especially work and task flow, in detail—**Critical**
- Champions the users' perspective and needs—**Critical**
- Knows business goals and desired business improvements, in detail—**Critical**
- Thinks strategically and yet is grounded in detail—**High**
- Has significant influence and authority within the team—**High**
- Has high-level skills for developing visual designs of screens, such as wireframes—**Critical**
- Creates fully realized use cases—**High**
- Knows the science of usability—**High**
- Understands programming well enough to persuasively defend user strategy—**Medium**

5. UI Developer(s)

- Has high-level display code skills—**Critical**
- Has solid graphic design skills—can "extend" existing UI or faithfully follow company-wide UI guide lines—often partnered with other UI Developer(s) who have highly developed display code skills —**High**
- Has cross-discipline experience in user experience design—can develop visual designs of screens, such as wireframes—**High**
- Understands programming well enough to persuasively defend user strategy, user experience, and the proper integration of display code—**High**

6. Project Manager

- Pro-active about the project—**High**
- Integral to "selling" the user strategy and user experience to the programming team—**Medium**
- Project Manager for entire development team—not just Project Manager for programming team —**High**

7. Team Leader

- Brings the development process to an intense focus on the user—**Critical**
- Champions the best mix of business, IT, and user opportunities—**Critical**
- Envisions, articulates, and evangelizes the core application user strategy—**Critical**
- Deeply involved in the user experience and user interface development process—**Critical**l

Best Practices Determining Business Requirements

1. Take the opportunity to improve and streamline business processes for the user—**Critical**

2. Perform work- and task-flow analysis with actual end users, managers, and stakeholders—**Critical**

3. Capture priority from a business point of view and frequency of use from a user point of view—**Critical**

4. Iterate with upper management, stakeholder(s), Business Analyst(s), operational managers, Team Leader and users until requirements are finalized—**High**

Best Practices for Gathering User Requirements

1. The team sets clear objectives for gathering user requirements, including timelines and success metrics—**High**

2. The user requirements-gathering process begins with well-defined user segments—those core user group or role types who will need unique features and activities in the application interface—**Critical**

3. User requirements are identified early on in the project, at the same time or before business and functional requirements are developed—**Critical**

4. Team members who will later develop user strategies, user experience designs, and/or user interfaces for the application directly participate in gathering and developing user requirements—**Critical**

5. User requirements are treated uniquely and distinctly, and are not simply a subset of business requirements, functional requirements, content requirements, etc.—**Critical**

6. The user requirements-gathering process is based on interacting with actual users who represent core application user types—**Critical**

7. The gathering process includes "contextual" interviews with actual users (one user at a time, at their place of work)—**High**

8. User requirements documentation is useful, minimal, feature-oriented, user type-categorized and consistently referred to throughout the development process—**High**

9. The methodology for gathering and documenting user requirements is appropriate to the project—**High**

10. User requirements are analyzed and prioritized by the entire team (or by key representatives of each role group in the development team)—**Critical**

11. User requirements include "how-it-is-now-done" and "how-it-really-ought-to-be-done" work- and task-flow analysis—**Critical**

12. The application development organization has a well-developed, ongoing user requirement gathering process—**Medium**

13. User requirements are gathered by team members who are trained, experienced, and skilled in user experience design methodologies—**High**

Best Practices for Designing User Strategy

1. The best teams make development of user strategy an actual development process "step" and perform this step in the beginning phases of development so the strategy can actually inform the development process —**Critical**

2. User strategies flow from teams with a strong, current knowledge base of user research, interviews, user requirements, and work- and task-flow analysis—**Critical**

3. User strategies are initially created from the ideal user experience perspective. Business requirements, IT programming concerns, scope issues, timeline issues, and budgetary considerations will be critical to make the strategies realistic, but they shouldn't dominate initially—**High**

4. A complete set of user strategies is created, one for at least each primary application user type and subtype (financial user, manager user, executive user, sales user, frequent user, infrequent user, etc.)—**High**

5. User strategies are at a high level, simply documented, kept distinct from any other interface details, and are developed before use cases and/or any interface details are worked out—**High**

6. User strategies are agreed upon by the entire team, and used as benchmarks for every specific user experience and user interface design decision made throughout the project—**Critical**

7. The user strategy is modified and refined as needed in the course of designing the user experience and interface, and throughout the life of the application tool—**Medium**

Best Practices for Designing the User Experience

1. First things first, teams should design, detail, and document the UE before moving on to UI design, to that degree they have a more efficient process and are able to focus UI issues when in that phase—**Critical**

2. Teams need to begin their development process with an intimate awareness of user type distinctions —**Critical**

3. The best teams conduct work- and task-flow analysis early on in the development project—whether or not they have a formal methodology—**Critical**

4. Create user type and subtype strategies—**Critical**

5. Bring it all together—design as a team, design in meetings and workshops, design with clear priorities —**Critical**

6. Develop the building blocks of a flexible and modular interface, including: navigation paradigm, application shells and grids, modules and module components, icons, action controls, wizards, widgets, success and failure messaging, help, and textual messaging. A highly modular system allows the most flexibility, and allowing for static or on-the-fly dynamic screen customization for various user types and subtypes—**High**

7. The best teams strategize early on in developing their content infrastructure, entry, and delivery systems —**High**

8. Design, create, and maintain a user experience and user interface style guide—**High**

9. Once globally used shells with navigation systems have been designed and proved, the best teams address work areas designed specifically for the various user types—**Critical**

10. The best teams show designs to users (UE designs, full UI integrated designs, etc.), whether formally or informally, in small groups or individually. They listen to the feedback and are willing to iterate their designs—**Critical**

Best Practices for Designing the User Interface

1. Optimum team make-up, skill sets, and processes—**Critical**

2. Give graphic design the latitude required to improve the user interface suggested by the user experience process—**High**

3. Know browser compatibility issues and the front- and back-end technical requirements and limitations established for the project—and/or interact intimately with development team front-end browser scripting specialists—**Critical**

4. Know the strengths and limitations of the development environment—**High**

5. Balance integrating or resonating with general corporate identity (CI) guidelines and customer-facing online branding requirements, with the usability issues surrounding longer-use internal applications—**Medium**

Best Practices for Documentation

1. Keep documents as simple as possible—**High**

2. Share documents in group meetings—**High**

3. Communicate priorities of importance as well as appropriate detail in all documents—**Critical**

4. Determine and document display code specifications at the beginning of development process—**Critical**

5. Translate business requirements and user requirements into features, and then prioritize them—**Critical**

6. Start as soon as possible developing visually-oriented documents—sketches, wireframes, etc.—**Critical**

7. Create a visually-dominated design specification—**High**

8. Begin creating (or modifying) a User Experience and User Interface (UE/UI) Style Guide as soon as the global visual design and global shell design are completed or updated—**High**

9. Critical best practice documents for developing optimum user strategy, user experience, and user interface design—**High**

- Requirements Gathering
  - > Business Requirements
  - > User Requirements
  - > Display Code Specification
- Feature Specification and Prioritization
- User Strategy
  - > Strategy Document
  - > Global Shell
  - > Global Visual Design
- User Experience
  - > Design Specification
  - > Wireframes
  - > Annotations
  - > Tables
  - > UE/UI Style Guide
- User Interface
  - > Visual Design Elements
  - > UI Standards (part of UE/UI Style Guide)
- Technical Specification

Joseph Selbie

Joseph Selbie is a founder and the CEO of Tristream, specialists in web-delivered application design. Extremely hands-on, Selbie's project roles include Team Leader, Business Analyst, interface designer, and workflow analyst. He is particularly adept at helping enterprises maximize the opportunity that new applications provide to increase productivity and ROI through improved workflow and usability. Tristream's work for Cisco's "I-deal" enterprise application was recognized by Nielsen Norman Group in their "Top Ten Intranets of 2001." Tristream's web-based application UI development process at major enterprises such as Cisco, KPMG, Adaptec, and Cendant is at the leading edge of the merger of traditional software development and emerging Web development processes.

Michael Coombs

Michael Coombs directs user interface design for Tristream. A former graphic designer and programmer, he understands development projects from back-end to interface. In the early Web years, Coombs hand-coded and designed corporate sites, and soon graduated to heading UI strategy projects for corporations, including Wells Fargo, Quovera, Identix, and Logitech. A devoted student of information design and human-factors research, Coombs quickly recognized the need for development team methodologies that integrate user research, and he brought these methods to Tristream in the mid-'90s. He was highlighted in Business 2.0 (Jan. 2002) for his e-commerce strategy work with Hooked-on-Phonics. For the last several years, he has focused on Web-application design for major enterprises, including Cisco (winning a "Top Ten Intranets of 2001" award from the Nielsen Norman Group), KPMG, Adaptec, and Cendant.

Methodology for the Best Practices Study

Tristream studied the development team practices of eight different development teams to identify the best development practices that resulted in superior user strategy, user experience, and user interface design.

In order to measure the success of the practices employed, Tristream conducted user interviews, in which users were asked to rank various aspects of the application. The user rankings provided benchmarks of success in usability and productivity for each application.

Applications that received high user rankings overall, or aspects of an application which received high rankings, indicated which teams were doing things well. It was these team practices that received the most attention.

The study was focused on identifying differentiating practices and methodologies. Practices observed, discussed, and defined were not limited to specific processes, though the majority of best practices we identified fit in this category, but also included company culture, quality and training of team members and any other factors that stood out or were identified as having had high positive impact by the team members interviewed.

User Interviews

After becoming familiar with the applications under study, Tristream interviewed from six to eight users from each primary user group of the applications, in order to evaluate their usability and productivity. Tristream interviewed users in from 1 to 3 primary user groups for each application. Interviews were conducted over the telephone. Users were encouraged to make comments in addition to making a numeric ranking. (See "User Interview" below)

Additionally, all members of the Tristream team ranked the applications by the same criteria to add a greater measure of objectivity to the results. The user rankings and Tristream rankings were averaged. The averaged rankings were then used to establish a best to worst ordering of the applications under study.

Development Team Interviews

Tristream then evaluated the development practices, team dynamics, and team composition of each development team. The development teams interviewed were internal, though in some cases partial outsourcing was used to help with the development of the user experience strategy, the user interface design, or code development.

Tristream interviewed development team members by telephone and on site. There was no specific set of criteria explored with each development team member (as there was with the users). Interviews varied considerably depending on the role of the team member involved and the practices that were revealed through each interview.

Tristream's evaluation focused on the differentiating practices of the development teams which "stood out" from those used by other teams and which resulted in significantly better user strategies, user experience, and user interfaces.

Given the subjective nature of the process, there were nonetheless surprisingly clear results. There were clear practices, team dynamics, and company attitudes that were present whenever there were superior results.

To further validate our findings, we ranked each team on their practices across 58 areas of development. We found a 1:1 correlation between the teams that employed the most practices identified as best, and the most usable and productive applications.

Not Addressed in this Study

Tristream's evaluation does not include many practices that have to do with quality assurance (QA), incremental change (bug tracking, bug-fixing, mid-release changes), user training, or the initial application launch. While these are practices that all organizations must learn and execute, they do not significantly affect the development of core user strategies, user experience, and user interface—the focus of this study.

Tristream's evaluation also does not address the strengths or weaknesses of any particular development system or technique (such as RUP, GPLC, XP), applications which aid development (such as Visio, Rational Rose), or development tools (such as UML). The best practices Tristream identified cut across all processes, applications, and tools.

This version of the best practices study for development teams concentrated on "core team" practices. In large application development projects there are frequently multiple teams working on the same application in large application development projects. In the next version of this report, Tristream will address the best practices for managing and supporting multiple teams working on the same application.

User Interview

Name:
Company:
Title:
Role (briefly describe):
Amount of time the application is used on average each day:

All questions will be ranked on the following scale from 0 to 6:

6—Completely Satisfactory
5—Mostly Satisfactory
4—Somewhat Satisfactory
3—Somewhat Unsatisfactory
2—Mostly Unsatisfactory
1—Completely Unsatisfactory
0—Not Applicable

1. Accomplishing Business Goals

a. The application enables me to perform all the major activities I need to perform in order to complete my work. _____
b. The application enables communication with others. _____
c. The application enables reporting and/or documentation. _____

General comments

2. Learning to use the Application

a. The application is very intuitive and it is easy to understand how to use it. _____
b. It took very little training time for me to learn how to use the application. _____
c. The application includes "help" features that enable me to figure out how to do things easily without asking others. _____

General Comments

3. Efficiency

a. Overall, I find the application to be very efficient. _____
b. I find that the application accomplishes tasks with the fewest possible steps. _____
c. I find the application provides me most of the information I need so I can accomplish tasks with the least amount of input from me. _____

General Comments

4. Flexibility

a. I find the application to be very flexible by allowing me to customize it to enhance the way I like to work. _____

b. I find the application to be very flexible by allowing me to save customized reports, data filters, etc. _____

c. I find the application to be very flexible by allowing me multiple ways to accomplish the same task. _____

General Comments

5. Navigation, User Controls and Search

a. Overall I find it easy to navigate within the application. _____

b. I learned to navigate within the application very quickly. _____

c. The navigation elements are visually consistent (similar kinds of controls look similar) and appear consistently in the same areas of the screen. _____

d. The navigation elements are always easy to see. _____

e. The application provides many "short cut" navigation elements such as drop down menu "quick choosers," and links conveniently located next to important data that help me to "drill" further into information. _____

f. "Search" is easy to use. _____

g. "Search" consistently finds what I am looking for. _____

General Comments

6. Information Presentation

a. The application presents information to me in a clear and easily readable way. _____

b. The application helps me to understand which information is most important. _____

c. Information-rich views in the application, such as tables, are easy to read and allow me to find specific information quickly. _____

d. The language used throughout the application is very natural and easy to understand. _____

General Comments

7. Accomplishing Tasks

a. The way the application works is well-matched to my own natural work processes. _____

b. The steps of processes I need to do in order to complete tasks are presented so that I easily know what I need to do to next. _____

General Comments

8. Integrating with Other People's Workflow

a. The application is very good at integrating my tasks with a larger group workflow. _____
b. I can easily get overall status views of various tasks and group workflows. _____
c. The application integrates communication, documents, and information from co-workers. _____

General Comments

9. Visual Design

a. The visual elements that make up the application are pleasant to use even after long sessions of using it. _____
b. The application allows me make visual preference settings. _____
c. The visual elements of the application help make the application easier to use by emphasizing important content and action controls, etc. _____

General Comments

