

# Tools to increase the strategic value of User Experience Design

James Nieters, David Grabel, Vijay Agrawal

SJC-J4, 255 W Tasman Ave, San Jose, CA 95014

[jnieters@cisco.com](mailto:jnieters@cisco.com), [dgrabel@cisco.com](mailto:dgrabel@cisco.com), [vijagraw@cisco.com](mailto:vijagraw@cisco.com)

## Abstract.

Case study describing tools and processes enabling accelerated adoption of Usability Standards, and increased efficiencies in development of accessible, internationalized, branded applications across large number of products in an enterprise. By building tools to support UE standards and best practices, the User Experience Team at Cisco not only achieved wide adoption of UE standards and best practices across Cisco applications, we also increased efficiencies in User Interface development and the ability to build internationalized, accessible software, thus increasing the strategic value of User Experience Design.

**Keywords:** User Experience, Branding, Tooling, Accessibility, Internationalization.

## 1 Introduction

Many experts in the field of HCI have talked about the value of human interface standards for graphical user interfaces (GUI's) [4] [3]. Such standards have been around since the early 1980's. Few, though, have discussed how User Experience Design (UXD) groups in a corporate culture can build tools to not only promote predictability and consistent brand experience, but to also drive a culture of usability, globalization and accessibility across application development teams.

The Cisco User Experience Design (UXD) Group built a set of application user interface standards in 2003. From the software development engineer's perspective, adhering to the standards required additional work- they had to read the specifications, interpret them, and spend time implementing hundreds of details [6]. While Cisco application development teams realized the value of consistent branding and design patterns, accessibility, and globalization, application features often received higher priority due to the high cost of compliance and time to market considerations [1].

In two case studies, application development teams only achieved full compliance with each of these requirements in the fifth release of the application. Doing so required three years of effort. Within Cisco, only three of over 100 teams developed entirely standards-conformant applications in the first year. This resulted in:

- Inconsistent branding
- Minimal accessibility and globalization support
- Increased customer training
- Increased support costs

The Cisco UXD Group recognized an opportunity to help application development teams solve this problem [2]. We created standards-conformant GUI component libraries and tools. The goal was to make it faster and easier to create a standards-conformant application than to build an application that is not conformant. The Cisco UXD Group achieved much greater adoption after having built these component libraries and tools. As of this paper, more than 20 application development teams use the UXD Components and tools, with more adopting them every week. Experiences thus far show a reduction in time to full feature implementation by up to 67%.

This paper discusses the process, technology and tooling choices made to achieve these objectives.

## **2 Risks and Benefits of Common Tooling**

The decision to implement tooling to support standards carries its own risks and benefits.

### **2.1 Implementation Risks**

Cisco has over 100 software applications. The risks of implementing tooling targeted toward a large number of applications in an enterprise include:

- The solution may not address common ground across a sufficient number of teams
- The solution may not be ready for consumption by teams when they need it
- Providing solutions can reduce innovation

#### **2.1.1 Building a solution that does not address the common ground**

For a tooling solution targeted to many application teams, it should meet the requirements of most teams planning to use the tool.

*At Cisco, this risk was mitigated by carrying out a detailed requirements gathering and analysis process of the applications being built at Cisco, and building a set of tools that supported as many teams as possible.*

### **2.1.2 Timing – inability to deliver the tool in the required timeframe**

The tools and feature enhancements should be made available to the teams at the appropriate time in their product development lifecycle.

*This was achieved by anticipating needs of the teams and working with them from product conceptualization through the UI design phase to ensure the necessary tooling was available by the time teams were ready to consume it.*

### **2.1.3 Innovation – balancing innovation and standardization**

Product marketing staff is tasked to understand customer requirements. Teams have been innovative in meeting these needs.

*A balance was achieved when development teams understood that they could contribute to our standards and tooling.*

## **2.2 Risks of not implementing**

Within Cisco, we learned that not implementing tooling as a vehicle to promote standards leads to:

- Increased time to market
- Increased support costs
- Added time to address non-English speaking markets
- Engineering inefficiency
- Inability to leverage advances in UI technologies due to lack of centralized core competencies.
- Inconsistent branding where we 'touch' customers every day: software applications
- Non-adherence to Section 508 (U.S. government regulations regarding accessibility)

## **3 Getting the Executive buy-in**

At Cisco Systems, Inc., ensuring adoption of standards required innovative designs and designs that meet business requirements. Adoption has also required involving politically powerful organizations as co-owners and aligning with senior-level initiatives. Initially, the organization started small by supporting the most strategic initiative in the organization—working with a cross-technology business council, for sponsorship and initial governance. The Cisco UXD Group focused to deliver value to teams in this initiative—standards-conformant designs in a timely fashion. The UXD

group produced examples of successful implementations, and worked with individual teams after this to sustain the momentum.

The biggest challenge was that while executives wanted to see teams adopt the standards, they would not require them to adopt the standards. Product teams can take on a “not invented here” attitude or view UE as a “speed bump”. The UXD Group’s response has been to approach this from two angles: Executive support and tooling that proved UE can speed time to market.

At this point, organizations that had not been on board are active supporters. In addition, we are closely aligned with Corporate Marketing, the Intranet standards team, and the Internet standards team. These collaborations give the UXD Group the added positional power it needs for the standards and tooling to be recognized by executives.

We continue to evolve the standards and tooling as the organization evolves. For example, we are defining a new icon language that reinforces the new Cisco brand, and will continue to evolve other assets as necessary.

## **4 Processes and Tooling**

This section describes the processes and tooling that facilitated wide adoption of the tools and standards, and thereby, increased the strategic value of the UXD team.

### **4.1 Processes**

The processes outlined below helped ensure we built the right set of tools:

- Early and continuous engagement with application teams
- A centralized standards and tooling team that works with Design Services to identify requirements
- Collaborative tools to help application teams provide insights and feedback to the tooling team. In our case, these included a wiki, jira, and email aliases.

### **4.2 Information Architecture Design**

Applications can be geared to different types of users, whether they be the first-time, novice or advanced. Functionality can be presented linearly (often for the novice user), logically, or based on usage patterns. Understanding the primary target audience has a direct effect on how functionality is presented. This is something that development teams seldom consider without UXD input.



**Fig. 1.** Output of Navigation Builder

It consists of:

- Navigation area– drawers or tree based navigation pane
- Content area– The area containing UI components and content, such as tables, reports, and charts

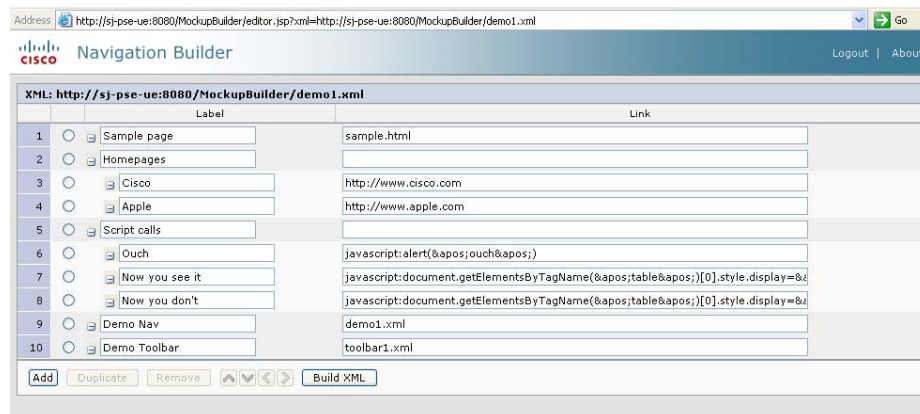
We divided prototyping tools into two parts:

1. Navigation Builder
2. Content Builder

We will see that using some of today’s tools, coupled with homegrown tooling, the UXD team achieved a robust prototyping platform.

The navigation builder enables designers to represent the navigation for applications via live prototypes. Designers use the following steps to use the navigation builder:

**Step 1: Visual specification of navigation elements**



**Fig. 2.** Navigation Builder front end

The diagram in Fig. 2 shows how designers enter navigation data. The value on the right defines the content link for each navigation item. For example, a label “Device A” would refer to the page devicea.html. The content link can be a mockup built using an IDE, a visio file, an image, or any other web accessible resource. This gives the interface designers flexibility to wire all types of mockups into the navigation.

**Step 2:** Internally, the navigation builder tool saves the navigation definition in XML format.

**Step 3:** To render the resulting mockup, the tool processes the navigation definition XML using a stylesheet and the stylesheet processor available in the browser.

The result is a standards-conformant application mockup with navigation and content areas presented to the designer.

An application prototype built in this manner is hosted on a shared server and can be accessed via a URL from anywhere. This supports easy collaboration among stakeholders of the prototype.

The Content Builder is a design tool enabling designers to build page content. Designers use tools such as Visio and Adobe Photoshop to build page content that is static. To prototype the interactivity of the next generation of UIs, a live mock-up tool based on Eclipse IDE is used. We built code templates, drag-and-drop wizard extensions to the Eclipse IDE to simplify mockup design:

- Code templates enable interface designers to start from a template. For example, a designer can choose to design a page containing panels of group boxes. This can be done by choosing the “Group Box Layout” template and specifying the number of group boxes when prompted. This results in the tool generating the necessary JSP tags using the SDK tag library and rendering them in the Design mode. An advanced designer may also choose the source mode and modify some of the html/jsp tag attributes to suit their needs.
- Wizards for complex widgets like table and scheduler enable designers to build the UI using easy-to-follow wizards, which reduces the need to understand or modify JSP tags.
- An added benefit is that the content builder generates code for the UI that can be used as a starting point by the application developer.

#### 4.4 UI Development Tools

These tools serve as building blocks enabling application developers to rapidly build standards-conformant UIs.

The broad goals for these libraries are:

- Built-in standards compliance
- Branded look and feel
- Reduced engineering cost for development teams
- Incorporation of latest technology advances into the tools, such as Web 2.0, AJAX-enabled interactive components
- Globalization support
- Accessibility conformance

#### 4.4.1 UI Component Libraries

There are over 100 software applications at Cisco. About half of them are desktop applications, and the rest are browser based.

UXD team decided to build JSP Tag library and Swing component libraries to meet the entire spectrum of applications.

JSP Tag libraries serve as the UI Components vehicle for browser-based applications. The team continually evolves these libraries to use the latest technologies such as AJAX, DHTML for rich, interactive look and feel. The standards define the visual behavior of the component and the tags implement them. This way, application teams automatically receive standards compliant, feature rich components

Swing components use a commercial component library and build a pluggable look and feel on top. This layer helps these components adhere to the Cisco UE Standards, thereby producing standards-conformant components ready for application teams to consume.

##### 4.4.1.1 UI Components to support Globalization

Supporting an interface for a global audience is more difficult than simply translating the text strings. It may include different number, date, and time formats, new input methods, redesigned layouts, different color schemes, and new icons [5]. Examples include:

- Text translations
- Orientation based on locale
- NumberFormat based on the client locale

Because internationalization issues do not readily emerge during development, teams do not notice them until close to the product shipment date. By this time, many such issues must remain unresolved due to release deadlines. The components in the UXD Group's library contain attributes to specify the locale and the key in the message bundle, thus supporting internationalization.

##### 4.4.1.2 UI Components to support Accessibility

Developing accessible UIs requires a sound understanding of accessibility requirements. The UXD Group worked with the Accessibility team at Cisco to help build UI components that comply with section 508 requirements. Examples include:

- Keyboard equivalents for all actions
- Appropriate color contrast and limits on animation
- Meaningful labels for tables and other objects
- Text Description for images

This enables application teams can build accessible UIs without having to spend effort on it.

#### **4.4.2 Globalization Tools**

A sustainable process to support globalization depends on many factors. These include leveraging translation memory and terminology databases, defining a taxonomy and globalization roles within the company, and establishing relationships with translation vendors in target countries. Although the Cisco UXD Group has helped promote this process, other teams own the non-tooling aspects of the Globalization program. The UXD Group has collaborated on the following elements:

##### *4.4.2.1 Translation Memory*

Many messages, such as error messages and tool tip text are repeated across multiple Cisco applications. For example, “Device x was successfully added.” Without translation memory, the same message is often phrased differently, leading to a non-uniform experience for users. Clearly, this central message library (translation memory) ensures that translations for commonly used messages are reused to ensure uniformity of messages, enhancing the user experience.

##### *4.4.2.2 Tools to detect globalization issues*

In the past, globalization issues were not discovered until late in the development life cycle, resulting in many internationalization challenges not being solved in time for product release.

The UXD team worked with the Globalization team to offer testing tools that as part of the development toolkit helped detect and remedy internationalization issues,. Doing so helped promote a culture of testing for internationalization issues as part of the day-to-day development process. The result is applications with robust globalization support.

#### **4.4.3 Splash and Login Screen Generators**

Splash and Login Screen Generators help generate images that application teams need for login screen, about, help and other places. A standard tool helps application teams quickly generate such images rapidly.

#### **4.4.4 Icon Libraries**

Icon Libraries are arguably the single most reused piece of software across Cisco applications. The UXD team offers a centralized repository of icons that helps application teams to readily integrate icons for various artifacts into their applications. It also ensures uniform look and feel. For example, an icon for wireless device looks same in all Cisco applications, facilitating predictability for an improved user experience.

#### 4.4.5 Help System

A uniform Help System design has helped application teams to quickly build Help pages and plug into their application using a common framework.

#### Summary

Developing tooling to help application development teams rapidly build globalized, accessible, and standards-conformant applications, has placed the Cisco UXD Group in a position of greater strategic relevance. Rather than functioning as auditors of usability and standards, the team functions as solution providers.

As of this paper, 20 application development teams use the UXD Components, with more adopting them every week. Experiences thus far show a reduction in time to full feature implementation by up to 67%. As teams adopt these component libraries, they are helping to build a culture of usability, internationalization and accessibility across the development community.

#### References

- [1] Bachman, Bill and Aliaga, Frederick– Four strategies for promoting common UI guidelines within Adobe - ©2003 ACM 1-58113-728-1 03/0006 5.00
- [2] Bazelmans, Rudy– Productivity – The role of the Tools group - ACM SIGSOFT SOFTWARE ENGINEERING NOTES Vol 10 No 3 Jul 1985
- [3] Grudin, Jonathan – Consistency, Standards, and formal approaches to interface Development and Evaluation – @ 1992 ACM 0734-2047/92/0100-0103
- [4] Myers, Challenges of HCI Design and Implementation – Interactions – Jan 1994
- [5] Russo, Patricia and Boor, Stephen – How Fluent is Your Interface? Designing for International Users – InterCHI, April 1993
- [6] Thoutrup, Henrik and Nielsen, Jakob – Assessing the usability of a User Interface Standard - 1991 ACM 0-89791 -383 -3/91 /0004/033
- [7] Jack Hakim, Tom Spitzer - Effective Prototyping for Usability - 0-7803-6431-7/00/\$10.00 © 2000 IEEE