

Components, Patterns, and Frameworks! Oh My!

By Jared M. Spool

Originally published: May 20, 2009

Somewhere, right now, there's a team creating a new design with some amazing, never-before-seen functionality. And to take advantage of that awesome, groundbreaking functionality work, their users will need to login.

Login functionality isn't new. It's not awesome. It's not very challenging to develop. Teams are designing this functionality as if it's never been built before.

But it has been built before. Teams, all over the world, have built login functionality into their applications about a million times. And yet, here we are, doing it all over again.

All this re-creation and re-invention isn't just inefficient, it leaves the team open to problems. Because it's not the sexy part of their project, it's likely to get less attention, resulting in an unusable and frustrating experience.

This is where the Re-use Trinity -- Patterns, Components, and Interaction Design Frameworks -- come in.

Teams are getting smaller. Many are half the size they were ten years ago.

Yet organizations are expecting more from each team. The projects are more sophisticated than ever. Agile development methods -- all the rage (and for good reasons) -- now force teams to think about using what's been done before. Re-use is the new priority.

We're seeing that the best teams have a re-use strategy that they divide into three types of libraries: Patterns, Components, and Interaction Design Frameworks. These libraries give the team speed and efficiency, letting them leverage the rich history of things-implemented-before.

In our research, we've found that teams that build out a re-use strategy see tangible benefits: They are more likely to get a completed design sooner, with all the little nuances and details that make for a great experience. Their designs are more likely to meet users expectations by behaving consistently across the entire functionality. Plus, the teams iterate faster (always a good thing), giving them a chance to play with the design while it's still malleable.

Patterns, components, and interaction design frameworks each play a different, essential role in making re-use work for the team.

Patterns - A Catalog of Desired Behaviors

Patterns were the first to come on the scene, inspired by the architectural patterns of Christopher Alexander. Alexander looked at the specific behaviors of how people lived and worked, and created re-usable descriptions of how a building's architecture could support those behaviors. The pattern didn't lock the architect into cookie-cutter designs, but instead gave them a resource to ensure they got all the details right.

Today's design patterns are similar. For example, let's say our user wants to enter a date while booking a reservation. What are the different designs that could support entering a date? A free-form text box with a parser? Three numeric tumblers, representing month, day, and year? A calendar pop-up, where the user just points and clicks?

Each option represents a design response to the same behavior. When the team specifies the response that works best for them (and their users), they can codify it in a pattern. Future teams, needing to respond to that desired behavior, can now respond similarly, meeting established user expectations while leveraging the previous work.

Many teams, starting their library, often turn to one of the off-the-shelf pattern libraries that are popping up on the scene. While these are often well documented and inexpensive (some are open source and free), they turn out to be less helpful than it would seem. This is because they are generic solutions, not taking the project's specific technological constraints and business requirements into account. The most helpful pattern libraries make these constraints and requirements their focus.

While we were off creating our pattern libraries, there was another move afoot: The developers needed an easy way to re-use specific code.

Once we choose the design response to our behavior, we need to start thinking in terms of the actual implementation. If we're to create a pop-up calendar, we have to make it work. Dates have to appear on the screen. The calendar has to respond to the mouse clicks. It needs to look like it is part of the rest of the application.

This is where components come in. Components specify the design response to the pixel level. Because they often are represented by their code, they also embody the specific interaction behavior. They contain the brand styling elements, such as the fonts, color, and look.

Developers use these components to piece together the specifics of the design. Once built, they are ready-made elements that can easily plug into any new screen. This speeds every part of the development process, from early prototypes to final deployment.

Interaction Design Frameworks - Putting the Puzzle Together

The Interaction Design Framework, conceived by designer Robert Hoekman, Jr., is the newest member of the trinity. Whereas components delve into detail of each pattern, frameworks collect groups of patterns to assemble a bigger picture.

Frameworks describe entire subsystems of patterns. For example, a login subsystem needs a pattern where users enter an id and password. But it also needs a pattern for password recovery, a pattern for setting up the id initially, a pattern for creating new ids, and a pattern for changing the password.

Teams create the frameworks by looking at what other designs have done. They become a checklist, helping the team ensure they've all the right patterns to start their design.

Frameworks are a high-level of abstraction. They don't talk to the specific branding or design. Instead, that's filled in by the components based on the individual patterns.

The benefits of re-use don't come for free. Identifying the re-usable elements takes time and practice. Documenting the elements can be time consuming. And keeping the libraries up to date is an ongoing and demanding task.

Making this a trinity helps. The work can be distributed between the designers and the developers.

Because the components are close to the final implementation, it's common that the development team members manage this library. Meanwhile, since the interaction design framework is about the larger experience, we expect to see the design team members take charge here. As for the pattern library, that's often a joint venture between design and development.

While small organizations can keep the library up-to-date with minimal effort, larger organizations should expect to need full-time curators. The curator's role is to encourage team members to add new elements to the library, while making sure existing elements are kept up to date. Since the libraries are a shared resource, the entire team shares the responsibility for the updating work.

The division of labor prevents any one person from carrying the burden of keeping the libraries useful. It also has the side benefit of keeping the libraries ever present in the work life of the team members, reminding them of its availability.

Making Good Design the Path of Least Resistance

Once built out, the libraries can become a powerful resource for the team. They can inform the design process, speed delivery, and make good design the path of least resistance.

