During the past few years, I've participated in a couple of workshops and talks by Jeff Patton where I learned about **story mapping**. This is a hands-on way to model a theme or project and slice it into user stories. Jeff first wrote about it in a January 2005 *Better Software* article, "**How You Slice It**."

What I learned from Jeff is to start by gathering your development team and business stakeholders. Create personas to represent your various types and roles of users, then think about how each persona would use your system or feature. What would they do first? What would they do next? Make a timeline of user activities using index cards on a wall, a table, or a floor. Then, go back and look at each user activity in detail and create user tasks and details about those tasks, which will eventually become user stories. Write those on cards, too, and stack them vertically under the corresponding user activity.

Once the story map is in place, you can slice them into user stories and plan which ones go into each iteration and release. You should walk the story map with stakeholders and see if you can think of any other details or issues. The story map helps you think about the value the system delivers.

Eager to Try It

Like most teams I know, our team struggles with getting the right level of detail on requirements for each user story before we start testing and coding. We don't want big design up front, but we don't want to waste a bunch of time going back and forth to the product owner and other business experts to nail down specifications during development. Brainstorming techniques such as mind mapping have helped us, but we still feel we lose too much time with requirements churn.

I've been agitating for some time to try story mapping and see if it might help us flesh out details about a theme and its user stories in advance of coding. I was pleased one recent afternoon when our ScrumMaster told me, "We want to story map the 'We Help You Choose' theme tomorrow."

I was thrilled to finally get my wish, but I was also swamped. I was leaving for Agile Testing Days in Potsdam in a couple days, and we were in the middle of a difficult and busy sprint. I didn't have time to go back and study up on exactly *how* to do story mapping. I thought I could wing it. (Cue shark attack music.)

One thing I understood about story mapping is that all stakeholders need to participate. We're often missing important stakeholders in our brainstorming and estimating meetings, so I insisted that all stakeholders for this theme must be present for our story mapping session. In this case, that was only one person, but many themes we do involve multiple stakeholders.**Doing It Wrong**
We gathered in our conference room with index-card-sized sticky notes, Sharpies, and a rolling whiteboard. Here's a little context about our domain: Our application automates all aspects of selling, creating, and managing 401(k) retirement plans. The "We Help You Choose" theme that we were planning revolves around a third-party vendor that helps 401(k) participants decide how to invest their retirement funds. If an employer providing a 401(k) plan to its employees wants to offer this service, the employer has to pay extra.

We had ten people in the room. The first thing I forgot is that you should divide into small groups of three to five people and have each group map the theme. Uh oh.

I started by asking our business experts about the personas for the theme. Who would be using it? The answer was the sales representative. Unfortunately, our stakeholders knew almost nothing about these sales reps, so we couldn't really flesh out the personalities of this persona. This was not a good start! All we could do was call him "Sammy Sales Rep" and say he was interested only in making the sale.

Next, I asked everyone to get up and grab a permanent marker and some sticky notes and start timelining the user tasks for this theme. What would Sammy Sales Rep do first? What would he do next? We knew the theme revolved around establishing a new 401(k) plan, so we would organize the user tasks around that.

Nobody got out of their chair. If only I had remembered about dividing into smaller groups! I repeated my request more assertively. The ScrumMaster got up and came to join me. Nobody else moved. In hindsight, I should have switched from doing the story map on the rolling whiteboard to doing it on the conference table, but I didn't think of that.

Giving up on getting people to physically move, I asked, "What's the first thing Sammy Sales Rep would do?" He's going to create a 401(k) plan for an employer. This is a multi-step wizard, so the timeline seemed fairly obvious. I wrote "Step 1" on a sticky note and put it on the rolling whiteboard. The sales rep would enter company information, then select contribution options for the plan, then select mutual funds or the "Help Me Choose" options, and so on.

Once we had this timeline of tasks, we went back and drilled into the details. What would change with each step compared with what sales reps do currently when they create a 401(k) plan? We noted the details needed for each UI page and some technical implementation requirements.

Unfortunately, we had started off at too detailed a level. We first should have identified "user activities"—the high-level things users do to achieve a particular goal. Establishing a 401(k) plan is an activity of the sales rep user. Managing the fund lineups is a user activity. We started with "user tasks"—specific steps within a user activity. "Select a fund lineup" is a user task within the activity of establishing a 401(k) plan.

Nevertheless, we got through a timeline of what Sammy Sales Rep would do. We also thought of the need for a user interface to manage the "We Help You Choose" program as well as a report. We also wrote stickies for a system to manage the "mutual fund lineups" that would be offered and for some reports to track which 401(k) plans have chosen a "fund lineup." We included many details about each step of the timeline, both from a user perspective and a technical-implementation perspective.

I also forgot that we should have walked through the story map with the stakeholders. This serves as a way of testing the map and finding additional details. Of course, now that we had the map, we could do that later.

What Did We Get out of It?

By the end of the meeting, I knew I had made several mistakes around the story mapping. I apologized to the team, threw my hands up in a "failure bow," and said, "How fascinating!" I asked them to applaud my failure. I hoped we would still learn something.

When the meeting concluded, I looked back at my notes on story mapping and found what I'd missed. I went back to the rolling whiteboard and added sticky notes for the missing "user activities" row. (We had implied user activities, but we hadn't specified them on the whiteboard.) I color-coded some of the details under each user task to reflect which were technical implementations, which were outstanding questions, and which were known tasks. Now, it looked more like a story map (see figure 1).

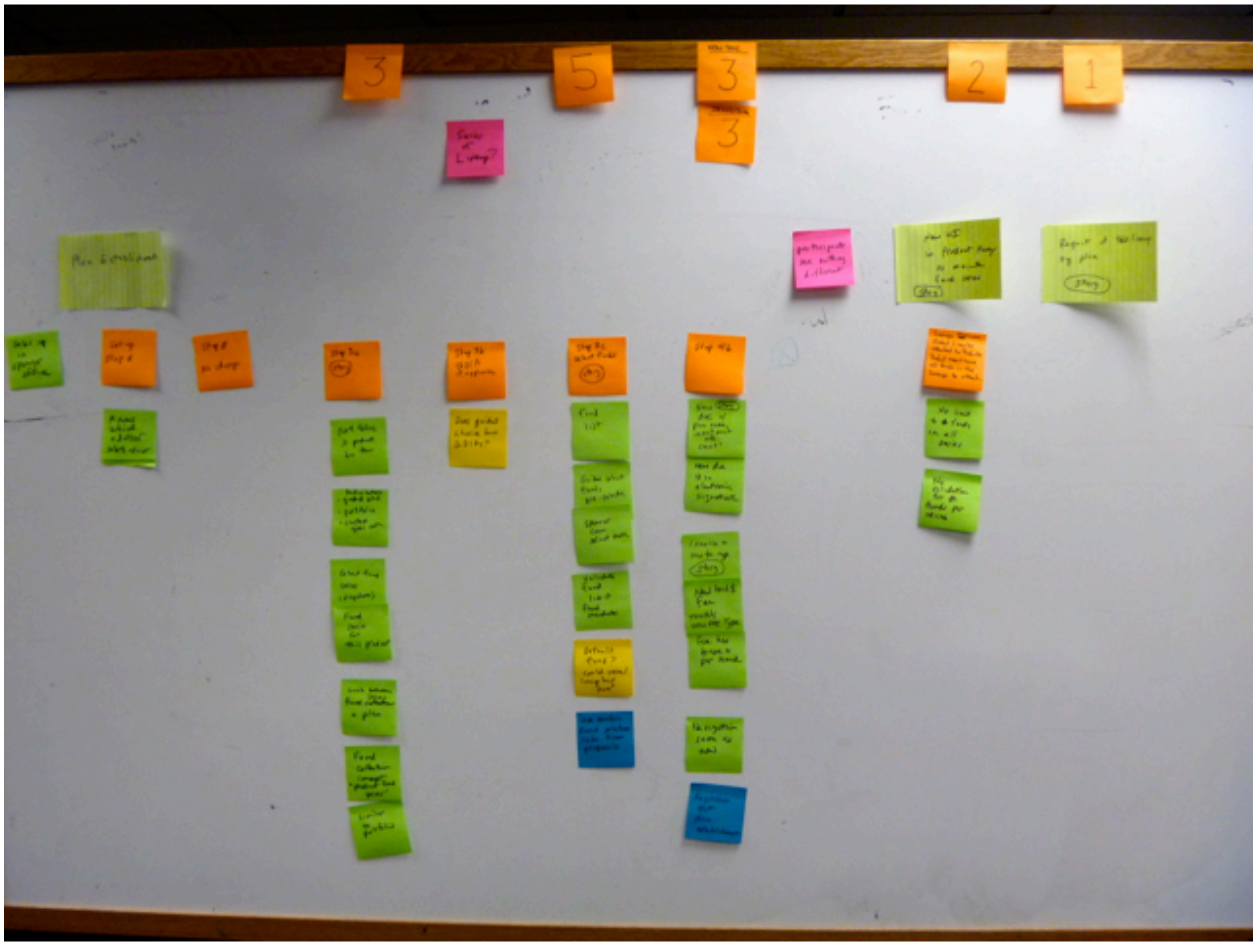

Figure 1: Story map with color-coded sticky notes

I talked with my teammates to get their feedback on our somewhat misguided "story mapping" attempt. Since they didn't know how story mapping was supposed to work either, they were OK with it. They felt that we had flushed out some details about stories in the theme that otherwise may not have come out until we started coding.

One programmer observed that he thought the story mapping was valuable because the stakeholder (I'll call him "Z") was at the meeting. Often, we only have the product owner to represent the stakeholders. Z turned out to be more receptive to simpler solutions than our product owner, and since he was the actual stakeholder, he got to make decisions. We got more information just from having Z participate in the meeting.

Next Time

We started working on the user stories for this theme in the very next sprint. We felt we were starting with a better shared understanding than we sometimes do for a new theme. We had sliced the theme into appropriately-sized user stories and understood the dependencies between them. However, confusion about part of the model for the underlying relationships cropped up that caused hours of wasted time and discussion. This is just what we were trying to avoid with story mapping, but it still occurred. That was frustrating.

In spite of still having some "requirements churn," my teammates feel that story mapping is worth trying again. They think it has potential as a brainstorming technique. Next time, we'll divide into two smaller groups, each will produce a map, and then we’ll consolidate. We’ll know to start with user activities and then drill down into user tasks and details. We’ll try mapping on the conference table, so I’m not the only one writing down the user activities, tasks, and details.

Even failures help us learn. I’m glad that I saw my mistakes and acknowledged them instead of getting discouraged. We improve by trying experiments, and the team is willing to continue this one. I’ll keep you posted!