

Improving the Obama campaign software by learning from users

Billy Belchev and Paul Baker

Webitects.com

billyb@webitects.com; paulb@webitects.com

August 25, 2009

Agile2009 conference proceedings

Abstract

In just a few days spent traveling to busy Obama campaign offices we learned enough from observing and interviewing users to propose substantial improvements and additions to the software that helps run campaign field operations. In this paper, we describe what we did and outline some techniques that your Agile team may wish to consider for your projects.

Background

The 2008 Obama campaign attracted record numbers of staff and volunteers, many of whom helped with day-to-day activities in local campaign offices. It also featured the greatest use of the Web—and software designed to run campaigns—ever in American politics.

One Tuesday in October, just a few weeks before Election Day (November 4), we received a call from Zack Exley, former Director of Online Organizing for the 2004 Kerry presidential campaign, asking if we could be in an Ohio campaign office in two days. Our task would be to study the software used to run the campaign’s “ground game,” with the goal of suggesting improvements that could be made before the next election cycle in 2010.

Zack, now at ThoughtWorks, had seen the results of design research for one of our clients and quickly grasped the benefits of such an approach for improving organizing software for political campaigns—and for nonprofits in general. He secured funding for the project and made the connections with campaign leadership and field office staff that allowed us to get unfettered access to local campaign operations.

To get an accurate picture of what really went on, we needed to go into the heat of the battle, so we traveled to three

“swing” states, five cities, and seven offices. We spent a total of nine days in the field over the course of five weeks.

We would soon learn that staff organizers and volunteers spent much of their time talking to undecided voters, in person or on the phone, trying to persuade them to cast ballots for their candidate.

Voter Activation Network (VAN) software helped them decide whom to call or visit, and how and when to do these tasks. The VAN stored millions of records about how people voted in the past, which candidates they favored in the current election, where they lived, and other demographic and contact information.

VAN developers were kept busy responding to top-down requests from campaign headquarters about data cleanliness and performance bottlenecks. It was our job to discover whether the software would also meet the needs of thousands of staff and volunteers on the ground, many of whom were younger and had high expectations about efficiency and ease of use based on their experience with popular Web applications.

We had many constraints: little time to prepare; a hard deadline (five weeks until Election Day); a geographically distributed user base; and a largely unknown domain space (except for Paul’s volunteer experience on local Chicago campaigns many years earlier). To succeed, we would have to be as Agile as possible.

What we did

It is very difficult to produce effective software unless the team tasked with its development has clear answers to several key questions: *Who* are the *user groups*? *What tasks* do they need to perform? *How* is each task done and in *what environments*? We demonstrate that these questions can be answered quickly and efficiently in just a few days learning from users.

Our main activities included contextual inquiry, lightweight usability testing, and artifact collection. We will explain how each was used, with examples.

The day after we got Zack's phone call, we packed our bags and boarded a plane to Ohio. We carried with us a portable HD camcorder with tripod and head; two pocket digital voice recorders; a small, high quality digital camera; two tablet PCs; a portable backup hard drive; a small laptop with Camtasia screen recording software; lots of back up batteries, pens and paper; and eyes and ears. And it all fit in our carry-on bags.

The value of context

Soon after we landed, Zack drove us to a local campaign office, which he had visited while working on an article for the Huffington Post [1]. He introduced us to one of the canvass coordinators he had met earlier, a 17-year-old high school student (we are calling him Adam), who agreed to show us what he did on the campaign.

Adam had suggested that we meet in a coffee shop where it would be quiet, with few distractions. We asked, instead, that he meet us in the campaign office. In our experience, you can often learn more by talking to one person in context than by talking to three people out of context.

The scene that greeted us appeared chaotic—people of all ages streaming in and out, phones ringing, field organizers rushing to meet deadlines, cold pizza sitting on card tables. As expected, this environment affected what organizers did and how they used supporting campaign software.

Adam showed his activities as a canvass coordinator, sometimes retrieving relevant pieces of paper, opening Google spreadsheets, or pointing out a person who influenced how he did his job.

In the coffee shop—without these clues—he might not have remembered to give us a complete view of what he did. We also would not have understood how busy and distracting the work environment was.

Contextual inquiry

An interaction such as the one we described with Adam, observing him as he did his work and asking questions, is called contextual inquiry (CI) and is one of the main techniques that we used.

Contextual inquiries are one-on-one interviews conducted in the user's workplace that focus on observations of ongoing work. [2]

Our approach to CI was necessarily very flexible, since we had little time to prepare and knew little about the domain we were studying.

We prepared no pre-written questions and did not ask each subject the same list of questions. Typically, we interviewed only one to five people, for 30–45 minutes each, in connection with a task.

Each CI painted parts of the picture, which helped us identify a next topic, interview subject, or line of inquiry related to the task we were studying. Interviews on specific tasks were interspersed with interviews about other tasks.

While learning about the process of canvassing, we might do two interviews in office 1, another one a few days later, in office 2, and one more, a week later, in office 3.

CI's related to tasks A, B, C, D, and E, conducted across offices 1, 2, and 3, could be represented on a timeline as |1| ABACE |2| BADDE |3| AC (see Figure 1).

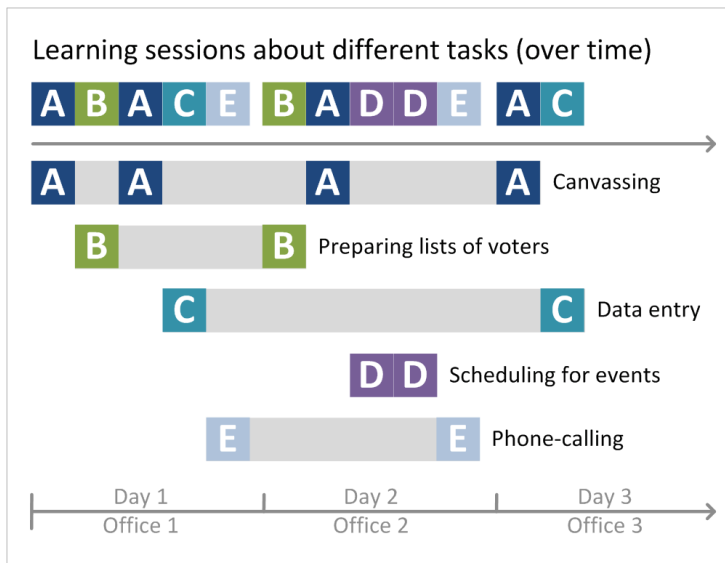


Figure 1. An illustration of how we spread learning sessions over time Each letter represents one session related to one task. It is more efficient and informative to intersperse learning activities from multiple tasks, as the opportunities arise, rather than trying to finish all activities related to a single task before moving on to the next.

In our first hour with Adam, we learned that *canvassing* and *calling* are the main organizing efforts and that many components of these activities are similar or identical, although canvassing involves more steps. Much of this work is done by volunteers who must be recruited and scheduled.

This knowledge influenced how we spent the rest of our first day, and subsequent days. For example, we knew that we would need to shadow door knockers as they “walked turf” in a neighborhood.

Lightweight usability testing

Our observations were not solely about the way things were normally done. For instance, if staff mentioned that they did not use a particular software feature that was meant to support their daily activities; we asked them to demonstrate that feature anyway, in order to learn why they did not use it. These “forced” demonstrations resemble usability testing.

The many varieties of wall calendars for scheduling and keeping track of volunteers led us to assume that the campaign software did not support these tasks. We asked a staffer about this and learned that it did, so we asked her to show us how. She walked us through it, but commented that it was too time-consuming, so they did not use it.

During these sessions, we captured users’ screens and mouse movements, asked them to “think aloud,” and recorded their comments. We also asked questions that arose from our observations.

Each session was 15–25 minutes, focused on one task, required as few as 1–3 users, and had no formal protocol (pre-screening, post-test, scenarios, debriefing, etc).

Items such as wall calendars for scheduling events are called “artifacts,” which we discuss next.

Collecting artifacts

Artifacts are copies or representations of physical or electronic “things” that the user creates, passes, or references to do a task [3].

Artifacts can be viewed as users’ attempts to improve the system, so whenever we see an artifact, our ears perk up.

Artifacts are easy to gather. We collected wall charts and calendars, canvass note forms, printed walk lists, calling sheets, precinct maps, driving directions, scripts for talking with voters, flip charts, campaign literature, door hangers, cheat sheets, and daily evaluations. We gathered, or photographed, more than 300 artifacts.

During Adam’s session, we discovered that he was using Google spreadsheets to store important information about *reliable* volunteers, which was not supported in the official campaign software. This turned out to be the case in all seven offices and, so far as we could determine, evolved independently within each office.

Another reason for the use of Google spreadsheets was that sharing them among members of the team required only that each person have a Google account. Organizers had tried Excel spreadsheets, which they passed around on flash drives via a “sneaker net,” however, volunteer information changed frequently, so these were difficult to keep current.

Multiple levels of permissions in the campaign software also limited access to other information that some organizers wished to share. To get around this, they exported the data and saved it to shared Google spreadsheets—or they shared system passwords, defeating the purpose of security.

Our copies of Google spreadsheets showed us the data that the staff used to effectively organize volunteers and informed the wireframes that we designed to support these tasks. Even when artifacts do not point out shortcomings, they often generate learnings that have implications for the system.

For instance, we gathered a few artifacts that supported social needs. In an office in one of the most Republican counties in Missouri, the first thing someone sees when they walk in the door is a wall covered with paper stars on which the names of all the volunteers are written (shown in Figure 2). Each star is color-coded by role.



Figure 2. A wall of stars in a campaign office in Missouri

These stars, bearing the names of volunteers and color coded by roles, greeted visitors. Besides telling who was doing what, they reassured people that they were not alone; that there was “safety in numbers.”

Besides helping keep track of volunteers, they reassure current and potential volunteers that they are not the only Obama supporters in the area (local Democratic Party candidates in this county refused to put Obama lawn signs next to their own).

Same day review sessions

In the type of environment in which we were working, there is no time to wait until you have collected all the data before looking for insights.

We held informal nightly sessions to evaluate what we had learned that day and what we wanted to learn the next. One evening, over warm beer and cold hamburgers, we decided that in the next day we needed to learn more about preparing canvassing or calling lists.

Another evening, we discussed an Ohio staffer’s comment at breakfast, “If a volunteer can’t make a canvass because she broke her arm, the next time we talk to her, we should ask how her arm is.” We began drawing parallels between a good volunteer management system and customer relationship management. We also began thinking that such a system could have important implications for keeping people politically active after Election Day.

We kept looking for more clues that might support or invalidate this need. After Election Day, we sketched out what such a system might do and how the supporting data might be structured.

During an Agile development process, quick review and analysis sessions can point out immediate fixes and improvements and help chart paths for further investigation.

In some cases, you might choose to include product owners or other stakeholders who have perspectives on the findings and ideas on how to address them. In our case, Zack Exley participated in some of our sessions and supplied domain knowledge that helped us more quickly understand the functions of various related pieces of software and the campaign apparatus structure.

Communicating results

Many Agile proponents advocate creating the least amount of documentation possible, and depending on direct team communication to fill in the details. However, we believe that Agile does not mean you cannot create simple, direct, creative ways of recording findings for later use. These types of documentation can also help get buy-in from stakeholders.

Immediate demands from the Obama campaign on software developers precluded our involvement with them during the few weeks before the election. We knew that to make an impact we would need to make convincing presentations to product owners, including the VAN and Blue State Digital (developers of Obama's public websites) after Election Day.

To serve this purpose, we combined Indi Young's Mental Models [4] with sequence modeling [2] to create a "gap analysis" diagram of our own (illustrated in Figure 3).

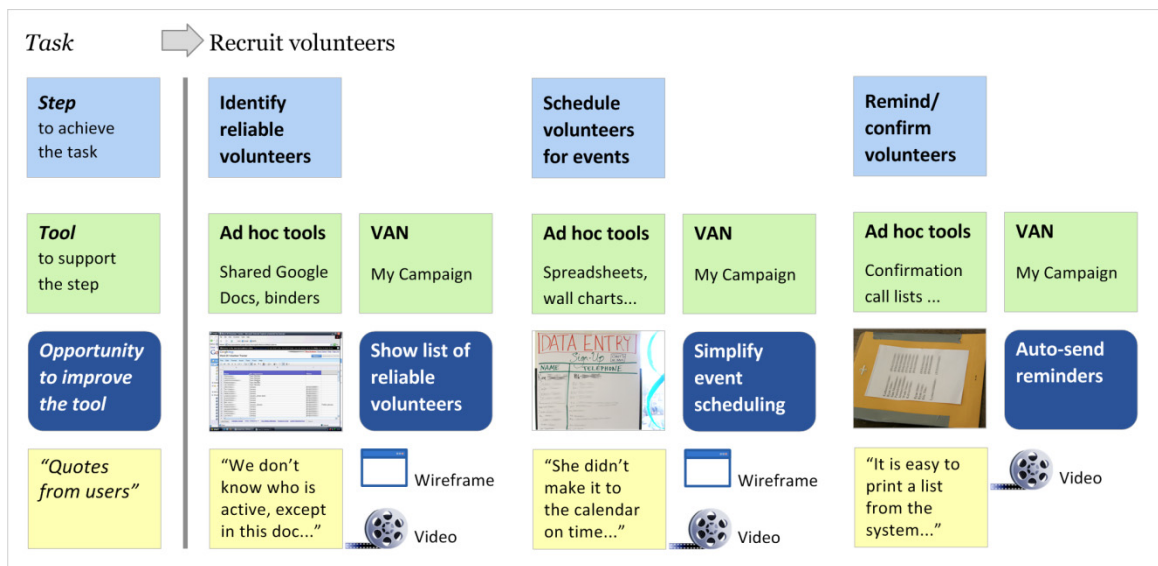


Figure 3. Our gap analysis model

Below the name of a campaign **task** are vertical rows of boxes related to that task. The **steps** required for each task are at the top of the rows. Below each step are: supporting **tools** (system or ad hoc/user created/artifacts); **opportunities** for improvement (dark boxes with white text); user-related **support** (comments, audios, and videos); and **wireframes** that illustrate possible improvements.

Nothing is more persuasive to product owners, business analysts, and developers than watching users struggle to use some part of their products. This is why we include audio and video clips that motivate improving important features.

The videos need not be polished—15 minutes spent editing a video in Camtasia can make a very strong point.

Delivering value

We made presentations to the VAN (primary vendor of campaign ground game software) and Blue State Digital (primary vendor behind barackobama.com and my.barackobama.com), which were greeted enthusiastically. Many developers and most of the companies' leadership attended, including company founders.

The president of one firm stated that he saw a lot of constructive new ideas, despite his initial reservations that we would only focus on shortcomings in the software. Another asked for copies of slides and wireframes for their UI staff, and

to use during a presentation to the Democratic National Committee the following week.

Of most interest were findings that revealed possible new features, would improve efficiency, and could help keep the movement going.

Discovering new features

Staff in all seven offices had some way of storing data about volunteers who, based on their histories of activity, could be counted on to help in a pinch. This was not supported in the official software, so organizers substituted Google spreadsheets, wall charts, and their personal phonebooks. We learned this on our first day with Adam and confirmed it in every office we visited.

We recommended that the system include tracking of reliable volunteers. Because we understood (from the interviews and artifacts) the tasks that needed to be supported and the associated data, we quickly designed wireframes to support this feature. Once we shared our findings and wireframes with product owners, they enthusiastically embraced this idea.

Improving efficiency

The presidential campaign attracted record numbers of staff and volunteers. This is not likely to be the case in future elections. The efficiency of campaign software may have a larger impact on their results.

Lightweight usability testing revealed inefficiencies that could easily be remedied. For example, staffers would save time if they could schedule volunteers from a profile *or* event listing screen. The first method was supported, but involved many steps; the second was not.

Keeping the movement going

Highly-motivated volunteers were essential to this campaign; they could also become the backbones of campaigns to support issues, win local offices, and change political power at the state and local levels.

Our findings generated ideas that could help shape the design of new types of volunteer management systems, develop software to support local activists, and create tools to facilitate the sharing of stories and ideas among organizers.

A few gotchas

Below are a few considerations to keep in mind.

People in power tend to give only the official point of view. People high up the ladder may idealize how systems are working. You will need to check what they say by observing and interviewing people actually doing the work. During this project, we learned that smart people, working on the ground, bent the rules and invented workarounds, to make work easier and more efficient.

Group interviews are often less useful than one-on-ones. We did not learn much from large group meetings, although we observed many. Users sometimes “parrot the party line” in public while, on their own, saying or doing something else. This is one reason to assure individuals of anonymity when observing or talking with them one-on-one. That is not to say that group meetings, interviews, or other activities (such as card sorting) are not useful in some situations—they just were not useful during this project.

Formality can kill the deal. The first time we interviewed someone on camera, we asked them to sign a consent form. While this is a common, recommended practice, it can get in the way of establishing trust. We learned that forms that required a signature tended to scare volunteers and staff. Some might have felt that they were giving us permission to publish the interview in the mass media, which the campaign did not allow. Although we stopped asking people to sign a form, we still explained why we were there and assured them of anonymity.

A few take-aways

Agile teams aim to deliver business value *faster*, but delivering *real* value requires a deep understanding of how businesses actually work. We have outlined an Agile approach to learning from users that is essential to gaining this understanding, and which can be done within short timeframes and in challenging situations. Below are some aspects of this approach.

Adapt, adapt, adapt. The activities that we have described in this paper are a means to an end. They assist decision makers, developers, and others on a product team in making better software. If, in furthering these goals, you change your mind, tactics, methods, preconceptions, or methodologies, so much the better. For instance, we abandoned our normal practice of asking people to sign consent forms when we discovered that it made them less willing to share.

Accept that you cannot learn everything, so prioritize. Decide which activities you do by considering their value. For example, you may not be able to interview people in all roles and at all experience levels. That is okay—prioritize based on who is most important within the context of what you are investigating.

Designers in an Agile development environment will often be working on user interfaces that are dependent on stories from several iterations. For instance, several activities can be going on at once: usability testing of recently released features; design wireframes for features currently under development; and learning activities to support features that will be developed in upcoming iterations.

Agile development depends on iteration or sprint planning, during which you choose which stories to implement based on priority, velocity (capacity), and dependencies. Since priorities can change quickly, designers need to be flexible enough to give input whenever it is needed.

Learning from users helps make difficult decisions. A short video clip of users performing a task or a review of related artifacts can often help resolve heated discussions such as which features to release or how to implement a specific feature. These reality-based arbiters help make decisions and garner support from stakeholders.

You can do this

The types of techniques and approaches we have outlined here can help a software team build better software by learning what *users really need* instead of what they—or product owners—*say they want*.

You do not need formal research training, just curiosity about what users do and how your products are being used.

Incorporating what you have learned into software development does not require all the techniques we have discussed. You can start with collecting some artifacts and conducting a few contextual interviews. Chances are, even with a tight project deadline, you can find time to do this.

Agile development is changing the way software is built. Integrating an Agile approach with learning from users is the next step in the revolution.

Acknowledgments

The authors would like to thank: Zack Exley and Judith Freeman at the New Organizing Institute, for this once-in-a-lifetime opportunity; Mark Sullivan and Jim St. George at the Voter Activation Network and Jascha Franklin-Hodge at Blue State Digital, for their cooperation and open-mindedness; and campaign staff and volunteers, for putting up with our constant questioning. We also thank Hugh Beyer for helping clarify our ideas and improve this paper.

References

- [1] Z. Exley, “The New Organizers, What's really behind Obama's ground game,” Oct. 8, 2008. [Online]. Available: http://www.huffingtonpost.com/zack-exley/the-new-organizers-part-1_b_132782.html [Accessed: May 20, 2009]
- [2] H. Beyer and K. Holtzblatt, *Contextual Design*. San Francisco: Morgan Kaufmann, 1998.
- [3] K. Holtzblatt, J. B. Wendell, S. Wood. *Rapid Contextual Design*. San Francisco: Morgan Kaufmann, 2004.
- [4] I. Young, *Mental Models: Aligning Design Strategy with Human Behavior*. New York: Rosenfeld Media, 2008.