



eXtreme DOCUMENTATION

BY CARL E. NUCKOLS, *Student Member, North Carolina State University Chapter*, and JEFF CANNA

A revolution is under way in software development, revolving around agile methodologies that allow more room for design changes based on input from customers during development. One popular agile methodology is eXtreme Programming (XP). When creating documentation in an XP environment, information developers must also allow more room for change during the process—a practice we like to call eXtreme Documentation.

Information development in an environment where the design specifications are flexible may not seem attractive at first. However, a comparison of software development methodologies and an examination of eXtreme Documentation strategies just might cast it in a better light. Three years ago the authors began working together on an XP project. At the same time Carl (an information developer) had the opportunity to develop comparable documentation for another project that employed a traditional software development methodology. The XP methodology had a great impact on how he performed his duties, and writing in this environment afforded him a chance to grow professionally in unexpected ways.

You may be drafted to document an XP project, or you may decide to introduce XP ideas into your company. Either way, knowing a bit about XP will keep you ahead of the game.

Comparing Development Processes

Traditional Methodologies

Those who have written software documentation for any length of time, and perhaps even those who have not been writing for many years, are probably familiar with traditional methodologies of software development. These methodologies

originated in the 1960s and were the gold standard of software development processes for three decades. They involve the following steps, with each step ideally finished before the next:

1. Determine and document the system concept.
2. Identify and analyze system requirements.
3. Divide the system into parts.
4. Design each part in great detail and write elaborate design specifications.
5. Code each part and test them individually.
6. Combine the parts and test the whole system in the validation phase.
7. Deploy the system.

With this process, information developers are usually presented with a detailed list of design specifications from step 4 which they use as a guide for writing the operator manual. They then have not only the coding phase in which to write the manual, but also a generally lengthy validation phase in which to document any changes to the initial specifications that occur during coding.

eXtreme Programming Method

A movement began in the 1990s to initiate faster, more flexible software development methodologies to better address the needs of customers. XP is an example of an agile methodology that balances the opposing forces of tight release schedules and software quality. It takes proven programming best practices—code reviews, unit testing, small integrations—and does them continuously.

Instead of a lengthy initial requirements documentation



process, XP uses customer stories produced in an iterative process. Customer stories are high-level requirements written in simple prose rather than the technical jargon common to some requirements documents.

XP also uses acceptance test scripts. These scripts are composed of commands that hook into the graphical user interface of the application being developed. They serve as an automated way to perform daily testing of sequences of user actions such as clicking buttons, counting rows in a table, and so on. The acceptance test scripts, which are written before the code that they specify, can replace the low-level design specifications.

Additional features of the XP process are the implementation of pair programming, in which software developers work in pairs, and a team environment in which the software developers, a “customer” (often a marketing person familiar with customer needs), and an information developer all sit in the same open room. Table 1 compares XP and the traditional methodology.

Initially, the XP environment seems an impossible place to produce good documentation. The process lacks the detailed design specifications document that is often used as a reference, and the test scripts reduce the time it takes to perform

validation to almost none. This situation appears to leave information developers doomed to work without a reference, while affording them almost no time to write using the completed application.

Developing Documentation in an XP Environment

There are strategies, however, that information developers can employ to thrive in this environment. Following are more details on key features of the XP development methodology and explanations of how information developers can take advantage of this methodology.

Open Team Environment

XP by definition fosters communication among all team members. Without this high level of communication it would be impossible to move at the pace required to develop timely software and documentation. Becoming part of the XP team allows information developers to better understand the application being produced and to offer the software developers additional insight into the application. Sitting with the software developers also affords information developers the chance to hear about changes to the application as they occur. This practice may sound impractical since most information developers are

responsible for documenting more than one application at a time, but they can still take advantage of this aspect of XP. The only items necessary are a laptop and a network connection in each area to allow them to rotate through their XP teams.

Sitting with the team also offers other advantages. First, it is much easier to get complete answers from software developers in this setting, because information developers are able to ask questions while they are working on the feature in question. With traditional methods it is sometimes difficult to garner complete answers, largely because information developers are asking questions about one feature while the software developers are engrossed in another.

Second, information developers can have much more influence in the XP process because of their proximity. They can leverage their understanding of the user and act as surrogate customers to help ensure that the customer’s needs are met. This is an important point, since it sometimes proves impractical for the team’s designated “customer” to remain in the room throughout a lengthy project.

Third, sitting with the team does a great deal to change the attitudes of the software developers. As evidenced by their posts on mailing lists such as TECHWR-L, information developers often feel that their contribution to the development process is not appreciated. Sitting with the team engenders an atmosphere of solidarity and cooperation.

Short Iterations

Another beneficial feature of XP is the implementation of short iterations. While rotating through their XP teams, it is imperative that information developers be present at the iteration retrospective and the planning session for the next iteration. These sessions give them frequent glimpses of what has just been added to the system and what is about to be added, so they can stay abreast of the project. Information developers may also possess the visual design skills and the understanding of user needs that allow them to offer suggestions about the design. The result is that the XP team can produce a better application, and

Table 1. Methodology Comparison

Traditional Methodology	eXtreme Programming
System employs a great deal of initial documentation.	System employs little initial documentation.
System employs a linear development process.	System employs a circular, iterative development process.
Software developers work independently on parts of the system.	Software developers work together as part of a team in an open environment with no code ownership.
“Customers” involved only in the initial phases during requirements gathering.	“Customers” involved throughout the process as members of the development team.
Information developers often located outside the development environment.	Information developers located within the open development environment.
Information developers often use detailed design specifications as a reference for writing the operator manual well into validation.	Information developers have no detailed design specifications to work from, and a short validation phase in which to complete the manual.

information developers can thereby further demonstrate their value.

Customer Stories

XP does not produce any significant initial documentation relating to user requirements. Instead, informally documented customer stories are produced for each feature. These customer stories are intended to be a reminder of a conversation between the “customer” and a software developer pair. Though the stories are not as detailed as design specifications, they are quite clear. By tracking the new customer stories being written, information developers know both what parts of the system on which to focus and the overall goal of each feature. They also know when to become involved in a particular conversation and what questions to ask.

Always-Shippable Application

A primary feature of XP is an application that is always ready to ship. Therefore, information developers can at any moment access a working application with all its available features fully functional. They can manually test what they are documenting while they write instead of relying on the accuracy of the design requirements document or the recollections of a software developer.

Acceptance Tests

The XP process also involves producing automated acceptance tests for a feature before any code is written. These acceptance tests use a scripting language that captures precisely how the user will interact with the system. The automated acceptance test scripts are the key to document creation in an XP environment for two main reasons. First, acceptance tests are an integral part of the XP environment, not just an exercise deemed necessary by someone outside the development process. Second, acceptance tests are designed to simulate a user’s interaction with the application. That fact makes these tests the logical reference for information developers, because they are generally more precise than any design specifications produced in traditional methodologies.

Information developers working in the traditional processes may receive the updated application only occasionally, so

they are forced to document from the specifications in an effort to stay on schedule. Unfortunately, they frequently find the design specifications are not as useful as they hoped. The specifications often do not reflect how the application eventually turns out. This situation sometimes results from problems in implementing features that cause the software developers to rethink the implementation of a function, and sometimes from the last-minute addition of new features. The result is wasted effort for information developers, who spend a great deal of time rewriting large sections of the manual. In traditional development processes, especially near the end of development, the specifications are sometimes updated to match the code after it is written, further delaying information developers’ work. Acceptance tests are more precise than design specifications and eliminate many of these pitfalls.

Conclusion

Information developers faced with writing documentation for an application produced with the XP development process are likely to feel some initial nervousness at the apparent disorganization of the approach. Many are fond of having design specifications documents as a reference and a quiet place in which to write. While the XP environment may seem foreign, it is important to remember that different is not always disadvantageous. Armed with the strategies just described, information developers can produce excellent documentation, and they are likely to soon find it easy to ignore the extraneous conversations. They may find creating documentation in the XP environment more rewarding than working in a traditional development environment if they follow two simple rules. First, information developers need to become active members of the team. They will learn from the experience, and so will the rest of the team. Second, information developers need to use the tools at hand. Acceptance tests are the key, so they should learn to decipher the scripts. If there are no scripts, they should take advantage of the open environment and ask questions.

The deciding moment in determining information developers’ preferred soft-

ware development process often comes at the end of the project. Most software projects end with a rush to implement last-minute features while still coming as close as possible to the target completion date. With projects using the traditional methodologies, software developers are often reluctant to spend time updating design specifications or making updated versions of the software available until they are through coding. This is not the case with the XP process, because the value of both the acceptance tests and the constant integrations prevent these practices from falling victim to procrastination. The result is a lower sense of urgency on the part of information developers near the end of software development to complete the manual. The information developers will also probably feel more confident that what they have written is complete and accurate.

Finally, information developers involved with XP may discover that they have developed skills for eliciting requirements and ensuring usability which prove useful in future projects. In the current economy, it never hurts to have a broader skill base. **1**

SUGGESTED READINGS

Auer, Ken, and Roy Miller. *Extreme Programming Applied: Playing to Win*. Boston: Addison-Wesley, 2002.

Tomasi, Martin, and Brad Mehlenbacher. “Re-engineering Online Documentation: Designing Examples-Based Online Support Systems.” *Technical Communication*. 46 (1) (1999): 55–66.

Carl Nuckols is an information developer in the information technology department at bioMerieux, Inc. He is also pursuing an M.S. in technical communication at North Carolina State University. He can be contacted at carl.nuckols@na.biomerieux.com.

Jeff Canna is a senior software developer with RoleModel Software, Inc. He has many years of experience creating systems and is very excited about the agile software development processes, especially eXtreme Programming. He can be contacted at jcanna@rolemodelsoft.com.