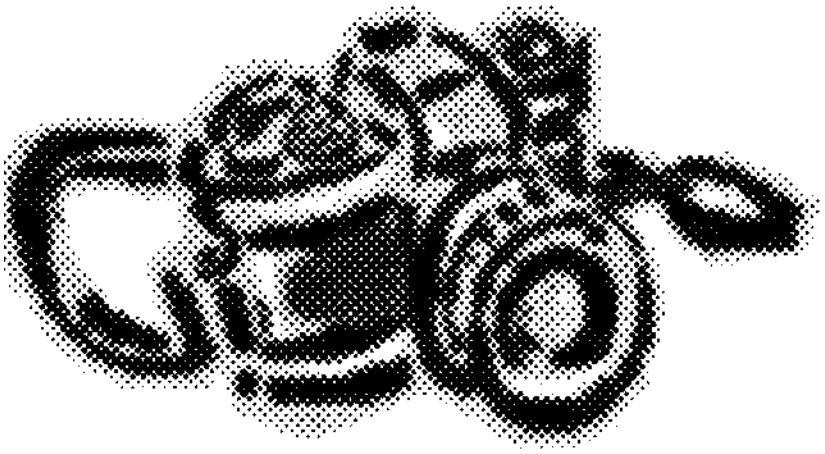


Make It Snappy!

THE STORY OF HOW KODAK WAS ABLE TO CUT LOCALIZATION CYCLE-TIME

BY SUZANNE TOPPING

If you deal with localization, chances are that you've heard this story before. The product-development team sits in a conference room at a gate review, and the topic of simultaneous worldwide product availability is raised. When some of the difficulties of achieving this goal are pointed out, one of the marketing team members mutters, "Oh, it's that localization issue again!"

Spoken as if localization is the thorn in the side of every project, existing only as a stumbling block to getting the product out the door. (Of course this was the same marketing manager who suggested that using a 24-hour clock would help eliminate the need for QA of localized software.)

Fortunately, most of the development team was more familiar with the process. Those in the trenches—project managers, software developers, QA testers, and documentation specialists—had all been around the block before. They realized that product development must be structured in a way that facilitates localization in order for things to happen quickly and efficiently.

The Challenge

It was with these people that the Kodak localization team worked in order to attempt the impossible: *simultaneous worldwide launch without English delay.*

Languages typically rolled out as shown in Figure 1. What was Marketing's reaction to this timeline? The localization process took too long!

In reality, the task of localization itself took only a very small piece of the total time. As you can see from Figure 2, during the final crunch from Gold English (i.e., final version) to availability of multilingual CDs, only a few days were spent updating localized versions. The bulk of the time was in testing and production.

The Solution

In order to move closer to simultaneous launch, the entire development process had to be examined. The process was broken into three stages:

- Software development
- Localization
- Multilingual-software QA

Since the localization time was already so short, the focus for change was primarily on the other two stages.

Changes to Software Development

For previous projects, the localization team gathered software components before handing them off to a vendor. The team gave blank specification forms to developers, hoping that the checklists would remind the developers to provide all required files, graphics, environment information, and build instructions along with the resource files. In most cases, however, the vendor still didn't receive everything they needed; build instructions weren't complete, header files were missing, or the wrong versions of tools were listed.

Resolving these issues delayed the start of the localization process by several days, which frequently derailed the schedule.

To reduce these problems, we created "Development Guidelines for Easy Software Localization." The information in these guidelines was based on common sense, but the act of writing it down somehow formalized and legitimized it.

Some of the most significant changes:

Isolate localizable strings. Kodak commonly received software back from vendors which contained strings (such as registry entries) that were translated when they shouldn't have been, and other

Figure 1

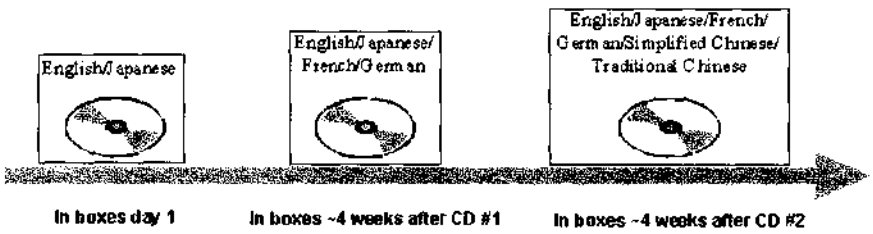
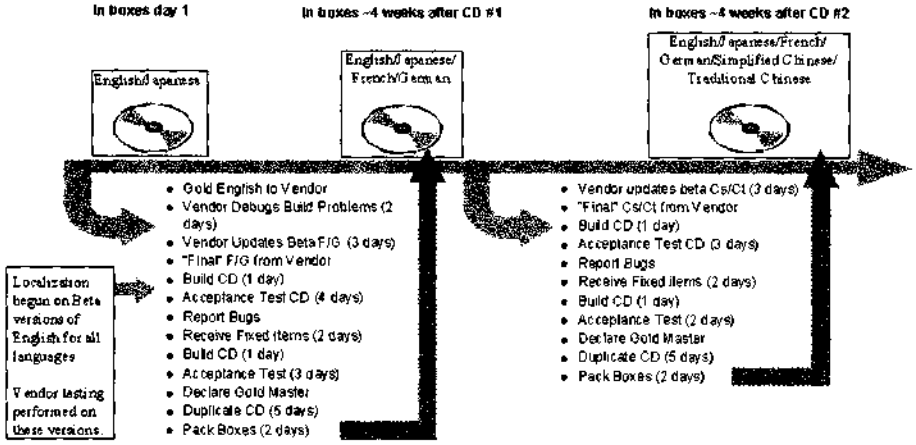


Figure 2



strings which were left in English when they should have been translated. This was not the vendor's fault, because no instructions were given about which strings to translate. The vendor had to sift through resource files, trying to figure out which strings in the code were translatable.

To combat this problem, Kodak decided that for each application, all translatable text would be pulled from the various resource files and placed in a single, text-string-only .rc file.

Handling the text in this way made it completely clear for the vendor. Everything needing translation was in one file, and none of the other files should be touched. It also helped prevent accidental changes to functionality, since no code was contained in the translated file.

Simplify build processes. The process of building software tended to be complex, requiring shared files and structures provided by other software developers. For example, one application needed a COM .dll to run, and that .dll was being built along with the rest of the application, rather than merely being provided for the application's use. Handing off this type of complicated build process to a vendor was a nightmare. The developers therefore agreed to review their build processes and pare them down to a more elegant form.

Document build processes. The development community also agreed to better document how builds were run. Not only did this help the localization process, but it also made it much easier to transition work between developers. Explicit instructions would be written so that anyone could sit down at a PC and build the application. Even a VP!

Pre-build before handoff to the vendor. As a final precaution, someone other than the developer would build the software before it was sent to the vendor. This meant missing files or other problems were identified before localization actually began.

Changes in QA/Testing

Once the front end of the process had been addressed, it was time to look at the back end. Testing of localized software had always been a challenge, even before attempting simultaneous launch.

Take for example the development of the DC220/260 digital camera in early 1998. The cameras were launched at the same time, used the same software CD and user's guides, but had slightly different feature sets. The full suite of test cases had to be run on both cameras. Figure 3 illustrates the test scenario for just one of the cameras. Before shipping the multilingual CD, this testing scenario had to be run for *both* cameras.

For Windows systems, each set of file folders in the figure represents an installer, four applications, three help files, and four read-me files. For the Macintosh, the folders represent a smaller set of files.

Given the large number of components, testing all languages simultaneously would require a huge lab and many test engineers. Since testing occurred only at project end, or roughly every six months, it didn't make sense to invest in these resources.

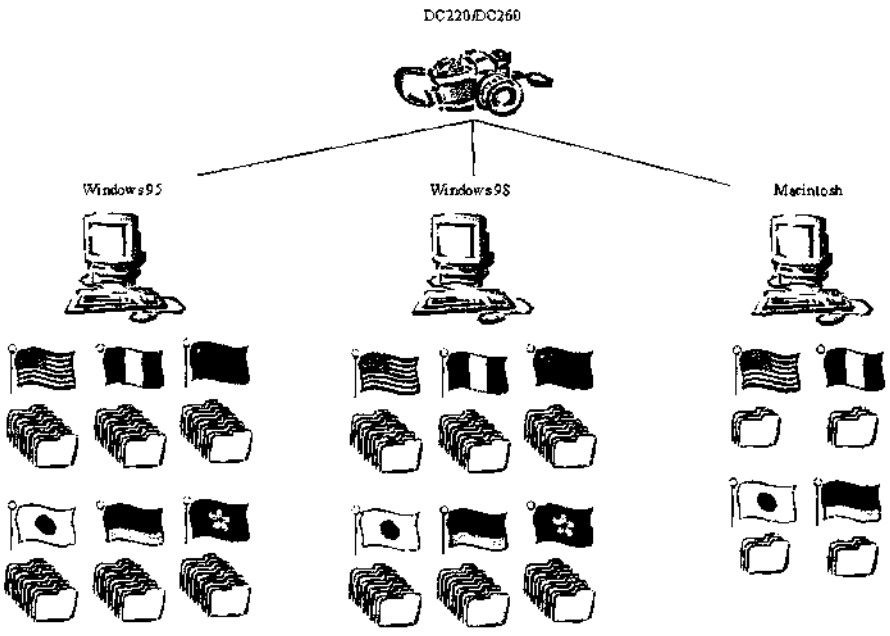
A Key Decision: Internal vs. Outsourced Testing

The first step was to investigate internal QA organizations to determine whether the testing could be distributed within Kodak. It was quickly decided that this approach was not feasible due to problems with camera availability, lack of linguistic resources, difficulty in obtaining localized third-party applications, etc.

The second step was to evaluate outsourcing. Throughout the last year, many localization companies began offering testing as a standalone service, so the timing was right.

To ensure a thorough evaluation, the localization, QA, and software-development teams participated in the process. Preliminary interviews were conducted by phone and email, and on-site visits were made to the most promising companies. The team eventually decided on a company that best matched the project needs.

Figure 3



Managing the Change

After deciding on outsourcing, the team realized that the entire paradigm for testing must change. Some of the issues identified were:

- Software specifications had to be thorough, and had to be finalized early.
- Terminology had to be agreed upon between Kodak, the localization vendor, and the testing vendor before testing could begin.
- Clear delineation of responsibility had to be established between what the localization and testing vendors would do. The localization vendor would perform *only* basic user-interface testing, and the test vendor would perform *only* functionality testing.
- Kodak QA staff had to evolve from being "doers" to being managers of vendor activities.
- A communication process for bug reporting had to be clearly defined so that information would flow smoothly between Kodak and the two vendors.

Given the tight timeline, communication was critical and accurate identification of who owned each bug was crucial.

Summary

Getting products out the door quickly is a requirement in today's competitive business environment. For Kodak's digital-camera projects, shortening the localization cycle meant revamping the product-development process.

But this principle is universal. Regardless of what you are localizing, putting better processes in place internally saves time and money externally.

About the Case Study

The above is an introductory excerpt from an extensive case study on localization at Kodak, and will appear in full in a book to be published in 1999 by John Benjamins in the ATA-monograph series. The book, *Case Studies in Language Management*, will feature real-life examples of language-technology and management techniques at global companies, large and small.

About the Author

Suzanne Topping

SUZANNE TOPPING spent 10 years with the Eastman Kodak Company, where she founded two localization groups. These groups managed localization projects for a wide range of products including thermal printers, digital printing kiosk systems, inkjet printers, digital scanners, and digital cameras. During her time at Kodak, Suzanne focused on localization process improvement and on educating project teams about localization needs. In October 1998, Topping launched Localization Unlimited, which provides consulting and training services in the field of localization. She is the author of *Graphic Design and Color in Today's Office* (1992).

Email her at stopping@rochester.rr.com