After much study and consideration, senior management committed to implementing a universal OO development methodology. Senior management recognized the need to improve customer satisfaction, which had always been high but needed to be better in an increasingly competitive market. Management saw the introduction of usability practices as a prime means to achieve this objective. So they began to place greater emphasis on usability—even to the point of building and staffing dual state-of-the-art usability labs.

Our goal throughout the projects we describe here was to combine the best OO analysis and design practices and usability techniques to create a powerful, unified way to develop software. We wanted user-centered design and evaluation to be a core component of the development process instead of an afterthought. Given the diversity and number of entrenched methods used within the company, implementing a universal methodology presented quite a challenge.

The company chartered a process coordination group to create the best possible process and to act as a change agent by educating and consulting with the various development groups. The process coordination group included the software engineering team responsible for the OO development process and tools. It also included the usability team responsible for the usability labs and evaluation processes as well as the company’s user interface standards.

Although the usability team made significant progress in introducing their evaluation process, they were frustrated to see it continually left until the end of the development process—at which point most changes are too costly to implement. The team wanted to bring usability into the earliest possible phases where it could have the most impact by improving initial design and eliminating rework.
The team members realized they had to integrate usability testing fully into the software development process rather than continue to support it as a complementary process.

A small group of representatives from the software engineering and usability teams took on the task of integrating user-centered design and evaluation into the company’s new OO development process. This task proved more challenging than we expected.

**Identifying the challenges**

The usability team promoted a user-centered design process that combined the contextual-inquiry design techniques developed or made popular by Hugh Beyer and Karen Holzblatt and the usage-centered design processes derived from the work of Larry Constantine and Lucy Lockwood. An overall design framework created by Charles B. Kreitzberg and Whitney Quesenberry guided the usability process through the software development phases.

The software engineering team chose the Rational Unified Process as the knowledge base to represent the company’s development processes. RUP is a comprehensive tool that provides information, guidance, templates, and examples for software engineering development activities. Unfortunately, although it acknowledges usability as a component of good software development, it inadequately supports such activities as the collection of usability requirements or usability evaluations.

So, one of our first challenges was to represent the usability activities within RUP. RUP’s process guide is customizable. However, because it employs extensive cross-referencing and provides for frequent updates, we had to execute modifications carefully to avoid difficult maintenance later on.

For the two teams involved, understanding each other’s processes, vocabulary, tools, and perspective was crucial. We spent considerable time trying to bridge the gaps, with each team participating in education classes sponsored by the other team. We couldn’t make real progress until we started to focus on the intent of each activity in the two processes. This let us move beyond both terminology and sequence differences. By happy coincidence, the focus on intent is also a key usability technique for identifying usability requirements.

To further complicate the situation, not all the development groups began using RUP immediately. In particular, some strategic projects used Princeton Softech Enterprise as the case tool for entering use cases and other OO artifacts. The developers used a variety of other tools, such as Visio and Caliber, for capturing the models, requirements, and diagrams created during the process.

Aligning these tools from a technological perspective and reconciling the differences in terminology presented further challenges. Both teams needed to be able to address developers, regardless of their background and tool experience. The usability team particularly needed to be able to explain how to integrate user-centered design into any of the development processes.

Dealing with organizational and cultural changes within the company proved to be a challenge equal to—if not greater than—the technical ones. Our older systems encouraged analysts to be more system-focused than user-focused. Business process modeling, or understanding our customers’ needs, although always a priority, really hadn’t been well structured. So, design decisions tended to put the burden on the user to learn a complex system rather than on the development team to produce an easy-to-use system.

Initially, these development groups became concerned that user-centered design in the early stages of development would lengthen the development process. Because user-centered design was so new to most people at the company, many had difficulty trusting this approach’s ability to reduce the overall development time by eliminating rework and improving design specifications. Clearly, user-centered design was not a natural way of thinking among many of the numerous development groups that shared responsibility for software products.

Until the development groups became accustomed to and proficient with these processes, they resisted implementing them. We determined that education and management support for the transition was vital.

**Agreeing on process**

Figures 1 and 2 illustrate the various activities in the software development process and the software development phases using the RUP nomenclature. The phases and steps don’t completely synchronize but are a general guideline. These figures also illustrate several key elements of the process:
The usability activities and their deliverables should emerge during the requirements gathering, analysis, and visioning steps. The better you execute these techniques, the better the product will be. Avoid the tendency to jump too early to coding.

You gather requirements, analyze them, and design the product vision iteratively during the inception and elaboration phases. The first iteration, during the inception phase, is a high-level requirements analysis for the project business plan. Later iterations, during the elaboration phase, are performed in a more detailed manner to produce the product design. Throughout all iterations of requirements definition, potential users provide insight.

The requirements and visioning activities feed both the system and user interface design.

Agreeing on a requirements process

All software development processes emphasize the importance of gathering requirements. However, many processes do not describe this step in detail. Ad hoc methods of defining requirements abound. Development teams commonly spend considerable time discussing and analyzing requirements. They also commonly argue about requirements late in the development process.

We recognized that the requirements process was the stage where both the OO development process and user-centered design process could have a major impact on improving the development process. So, we made requirements a priority and researched and analyzed requirements-gathering methods to select a best-practice direction.

The user role models—or profiles—described by Constantine and Lockwood are of great importance for understanding and interpreting the project requirements. Contextual inquiry provides an effective, detailed, and structured requirements-gathering technique. Defining the user and learning about the work process at the place where the work occurs, and then analyzing the findings in a structured way, can considerably reduce the time a company normally takes to define requirements.

We have divided requirements definition into four steps (see Figure 1):

1. Creating concepts. This step is the initial business-modeling process. High-level requirements gathering becomes necessary at a project’s inception to define the business plan properly, because the business plan is the main deliverable for this first step. To the business plan, we add a concept statement for the product that includes an outline of the intended end user, what the product should do, and the initial usability goals.

2. Gathering requirements. This step is best done through site visits to see the actual work taking place. After each visit, we capture the findings and collect them in detailed user profiles and a variety of models illustrating the observed workflow sequence, the communication pat-
terns, the artifacts used (such as documents and equipment), the cultural climate, and the physical environment. Beyer and Holtzblatt’s contextual-inquiry technique uses five models. We condensed these models down to a more flexible set that we can use as needed, depending on the particular product’s scope and activity.

3. **Analyzing requirements.** After an appropriate number of site visits, the requirements team interprets the data and compiles a user profile that represents the common traits of all the users observed and interviewed. The team also consolidates the workflow, artifact, cultural, and physical models. In addition, the team defines detailed usability objectives for the project and creates an affinity diagram of all the issues still needing investigation.

4. **Designing the product vision.** In this step, the team develops a model envisioning the final product’s strategy for meeting the requirements.

Beyer and Holtzblatt call the product vision the user environment design model, Cognetics calls it a roadmap, and Constantine and Lockwood call it the contextual model. The product vision provides the structural blueprint for the product and how the end user will interact with and navigate through the system. This model is a key component of the user-centered design process.

Developing essential-use cases with extended-use cases to illustrate requirements is part of this vision model; this enables evaluation of all stages of the design. By agreeing that development teams could use a combination of context diagramming, use cases, and storyboarding to create this vision model, we were able to integrate this key component into the software development process.

**Agreeing on design and execution processes**

With a good product vision well documented—both visually and with use cases—the rest of the software development process falls into place. Four steps remain (see Figure 2):
1. **Designing the user interface.** Building UI prototypes lets the team test its design with potential end users. Iteratively testing the design, refining it, and retesting it until the team is certain the design works ensures the product’s future success.11 You accomplish the testing in this step with a series of prototypes, starting with rough, hand-drawn paper sketches and ending with detailed mock-ups simulating the functionality.

2. **Modeling the system—analysis and design.** You map use cases with system responses to domain models; the business objects lead to OO models, including class, collaboration, sequence, and state diagrams.12,13

3. **Implementing UI design.** Now, and only now, is the product coded. All the requirements analysis, as well as the previous iterative testing and refinement of the design, ensures that the design specifications have greater detail and that this stage involves less rework. At this point we hope to see the development teams recognize and appreciate the process’s cost-effectiveness, because they find that coding is more efficient than it has traditionally been.

4. **Transitioning the project and supporting the product environment.** These steps cover the product’s rollout and production. During production is the time to perform usability studies at user sites to both complete the cycle and begin the next cycle.

All software development process activities are potentially iterative. To mitigate project risk, iteration is critical to the project’s success. Some iteration must occur even within a cycle. Revisiting and reconciling the high-level models and design concepts created during a project’s initial steps is vital as a project moves into the construction phase. This activity keeps the project focused on its vision, within the scope of its requirements, and on track with its budget.

Flexibility in project development is also essential. The software development process we developed might seem a generic solution. But good judgment in adapting the process and choosing activities appropriate for any specific project and its time frames is always necessary.

**Usability roles**

While clarifying the process changes for our company’s developers, we identified some new roles and modified some existing ones on our development teams. A role doesn’t necessarily equate to a person. More than one person can perform a single role; a single person can perform more than one role. We based these roles loosely on those defined by Deborah Mayhew.14

The **usability engineer** has primary responsibility for gathering and analyzing user requirements and for expressing those requirements in the product vision and use cases. These responsibilities extend beyond those of the typical business analyst. The business analyst gathers requirements to determine what the product should do. The usability engineer studies the potential end users to determine how the product should do those things. This means collecting details about end users, such as their typical level of education and level of experience with computers, as well as details about their work, such as the average turnover rate and their criteria for performance evaluations. This role’s goal is to capture the end users’ mental model of the work.

The **user interface designer** sculpts the end users’ interactions with the product, developing the early prototypes of the product design. The UI designer also evaluates the usability of the design prototypes. In addition, this role creates the design specifications. This role’s goal is to express the end users’ mental model as closely as possible within the constraints of the information and technical architecture.

The **usability evaluator** has primary responsibility for testing the product design, analyzing and documenting the results, and presenting the results to the development team. The bulk of this testing takes place during the initial stages of design before you code the product.

The **user** has no primary responsibilities, but is key to the product’s success. Users help define the requirements and design the user interface. They also evaluate the UI during usability testing.

**Workflow models**

Along the way, we attempted to simplify the process. While consulting on a development project with extremely tight deadlines, one of the authors, a UI designer, found the number of models necessary for reporting field observations too time consuming. Because she still needed to document and share the knowledge gained from her field
studies with Unified Modeling Language-trained analysts on her team, she created a workflow model that condensed the Beyer and Holtzblatt sequence and communication models. She also designed it to resemble UML so that the analysts would understand it more easily. It was a success.

We incorporated the new workflow model into our interpretation session models, which drive the analysis following site visits. Within one model, we show both the communication relationships (or transfer of data) and the activity’s sequence, which helped us reduce the number of models that we created to represent the users’ work. Figure 3 shows a simple workflow model from a site visit.

In Figure 3, stick figures represent the user being studied (in boldface) and other...
human actors involved in the task. Communication arrows extend from the principal actor to the other people (and artifacts) with whom that person interacts. We label each arrow with an action or message. The activity is broken down into tasks and task steps that we number to show the activity sequence. Each task’s principal actor is on the left, letting the model show where the user is a recipient of an interaction. At least one intent must be identified for each task.

When we move to the next stage of analysis, in which we consolidate the lessons of each field study (in preparation for designing the product vision), we study the intents from all the workflow models. Then we create a consolidated workflow model to depict the task structure and the strategies common to our various customers. The other contextual design models—artifact, cultural, and physical—are consolidated as well, and we build an affinity diagram of the issues and insights from the field studies. Figure 4 shows a consolidated workflow model from another set of field studies pertaining to patient bed location.

Reaching agreement early

Reaching a high-level agreement on the essential processes was the first step toward integrating usability into our company’s process. Once we achieved this agreement, working through the process details became easier. We chose to use key strategic projects to incorporate the main techniques before rolling the process out to more established development teams.

We have kept in mind that rolling out a company-wide process needs to be done over time. Upper management must stand behind the process, and the development teams must also buy in. Maintaining the big picture is critical to a successful implementation.

We found that teams need to focus on several steps early in the process:

- Refine the product’s user profiles to ensure that the entire team has a thorough understanding of the users.
- Prioritize site visits to gather usability and functional requirements.
- Have key usability engineers provide thorough and structured interpretation of the data collected from site visits.
- Build the software blueprints—the user environment designs—and provide roadmaps to guide the rest of the process.

The user interface designers must be versed in usability principles and employ these principles as they work. Also, usability testing must take place during the early stages of design.
We embrace the observations of Al-istair Cockburn, who stresses the people side of software development. He points out that the human factors have dominance over any other factor and that the development process must consider the human factors within the development team, as well as those of the end users.

No matter what our company’s ultimate software development process turns out to be, it must address Cockburn’s very real 14 tenets, including these four:

- People act according to their reward.
- Recognize the presence of dominant personalities.
- People work in certain ways better than others.
- The communications load can soon dominate the project.

Until a company understands, accepts, and finds ways to address Cockburn’s tenets, every software development process will face challenges that could be more easily solved by applying some of the principles we’ve discussed here.

We’re not suggesting that usability techniques are a panacea to every software development ill, but we are making an appeal to implement some of the principles we’ve discussed here.

No product will ever be “perfect.” And eliminating rework completely will of course never be possible, because—after all—software development is almost endlessly iterative, with shifting user needs and the resulting requisite upgrades. But we can still measure several benefits in very tangible terms: user satisfaction. Injecting the user’s voice early in the process is our main objective.

References