Constantine & Lockwood, Ltd.

PREPRINT

# Users, Roles, and Personas

Larry Constantine, IDSA
Chief Scientist, Constantine & Lockwood, Ltd.

*You cannot think outside the box when you are trying to represent a box.*
Brian Hayes

To do effective design you need to understand your users and their needs. While no credible school of thought in design would take serious exception, opinions vary considerably on how best to gain that understanding and how to record and communicate it once you do. Models of one form or another are the medium for the message in most design methods. Models function as intermediaries between the often ambiguous, overwhelmingly complex reality of actual users and the more narrowly focused and specific needs of designers.

In usage-centered design (see sidebar, "Users or Usage"), user roles capture and carry the essential understanding about users. User roles, one of the three core models of usage-centered design, are close cousins of personas but differ in a number of ways of potential significance to designers. In an attempt to sound a counterpoint to complement the main themes of the book, this chapter introduces user roles in the context of usage-centered design and explores the relationships between user roles and personas. Compared to typical personas as presented in this book and elsewhere, user roles are a more compact and concise representation that is more finely focused on issues with direct relevance for visual and interaction design. For these reasons, user role models can also be simpler and faster to develop. Although roles and personas can complement each other, user role models may also, under some circumstances and for some purposes, offer distinct advantages for designers.

The connections between user roles and personas are not entirely accidental, as usage-centered design and goal-directed design share similar albeit not identical philosophies of design and a history of mutual influence (Constantine and Lockwood, 1999; Cooper and Reimann, 2003). Both approaches are organized design processes that emphasize fitting the interaction design to the genuine needs of users. Although probably not as well known as goal-directed design, usage-centered design is a widely practiced alternative with a decade-long track record for producing innovation and breakthroughs in user-performance, particularly in complex applications where efficient and dependable user performance is critical, such as in medical informatics (Strope, 2003) and industrial

## Users or Usage

The distinction between usage-centered design and user-centered methods is, as the terms themselves suggest, a matter of emphasis rather than an absolute difference (Constantine and Lockwood, 2002; Constantine, 2004a). Whereas user-centered design makes users per se the center of attention and seeks to promote user satisfaction with the entire user experience, usage-centered design is more narrowly focused on user performance and on the creation of tools to enhance the efficiency and dependability of user performance. Although both approaches combine field study and user involvement with modeling, in usage-centered design the models are in the foreground with user studies and user involvement in the background. This difference in emphasis can lead to differences in outcomes. Indeed, it has been argued that over-dependence on user feedback and involvement in user-centered approaches can discourage innovation and contribute to unnecessarily conservative designs (Constantine, 2004a).

automation (Windl, 2002a). Usage-centered design has also been influential in shaping other methods, most notably user experience modeling (Heumann, 2003) within the Unified Software Development Process (Jacobson, Booch, and Rumbaugh, 1999; Kruchten, Ahlqvist, and Byland, 2001), which borrows and adopts core usage-centered modeling techniques first introduced into object-oriented software engineering (Ahlqvist, 1996), the methodological precursor of the Unified Process.

In usage-centered design, models guide the designer throughout the process. As represented schematically in Figure 1, the final visual and interaction design derives more or less directly from a content model or abstract prototype (Constantine, 1998; 2003; Constantine et al. 2000) that models the content and organization of the user interface independent of its detailed appearance and behavior. The content model is itself based on a comprehensive task model expressed in the form of so-called essential use cases (Constantine, 1995; Constantine and Lockwood, 2001) or task cases. (See sidebar, “Use Case Essentials.”) Task cases, in turn, support user roles as represented in the user role model. Even the initial investigation and data gathering, which provide the information and insight needed to build user role and task models in the first place, is a model-driven process based on exploratory modeling (Windl, 2002a). In this process, provisional modeling of user roles and tasks is used to generate questions and issues that help focus field work into a faster and more effective inquiry process.

**Users or Usage continued…**

The emphasis on models and modeling in usage-centered design evolved over time from the need for a design process that was both easy to learn and practice on the one hand and, on the other, predictably led to superior designs. From its inception, usage-centered design has been shaped by the goal to enable ordinary user interface designers to produce extraordinary results, to have a process that is less dependent on the skill and artistry of a few exceptional designers than on consistent application of proven techniques. In the interest of streamlining and systematizing the design process, the models of usage-centered design are condensed, simplified, and sharply focused on those matters with greatest relevance for driving the design forward toward the best results. Because of this in-built efficiency, usage-centered design has proved particularly compatible with short development cycles and modern agile development methods (Constantine, 2004b; Constantine and Lockwood, 2002; Patton, 2002, 2003). At the same time, usage-centered design is a fully scalable approach that has been applied to projects ranging from upwards of 50 developer-years (Windl, 2002b) to massive multi-year efforts involving hundreds of developers.

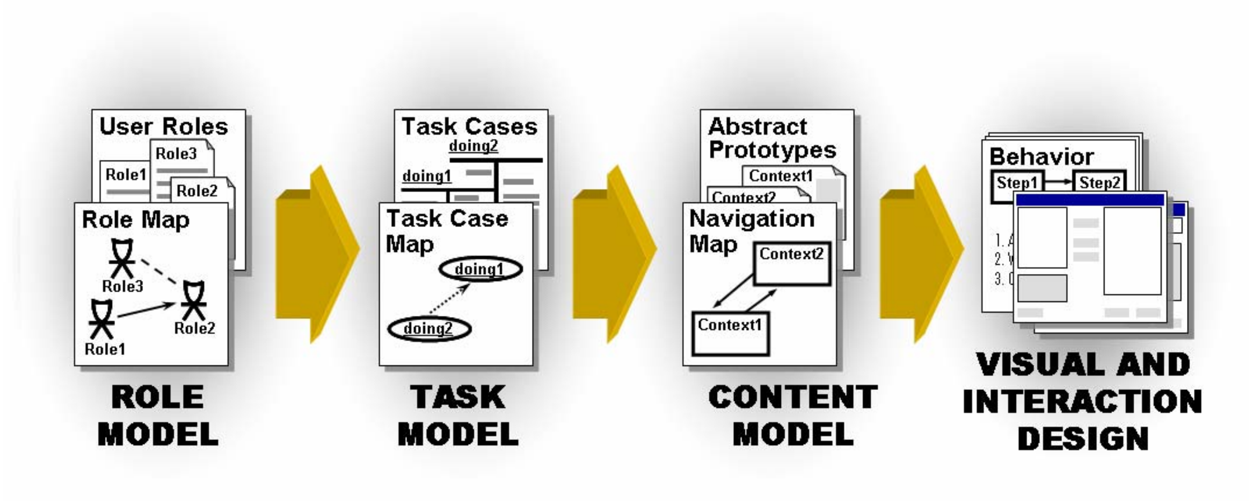


Figure 1 - Logical dependency of models in usage-centered design.

Usage-centered design began its evolution without a name when, in 1993, the author and Lucy Lockwood formulated an improved form of use cases to help solve a user interface design problem. Use cases are one of the cornerstones of modern software engineering practice, but in their original form (Jacobson et al., 1992) were not well suited for user interface design, having too many built-in assumptions and premature decisions about the user interface being designed. The abstract, simplified form of use cases created for usage-centered design came to be known as task cases or essential use cases (Constantine, 1994; 1995) and have become widely used in interaction design and requirements modeling (Constantine and Lockwood, 2001; Cohn, 2004). The need to anchor the task model in an appropriate understanding of users quickly became apparent to those working with essential use cases. User roles (Constantine, 1995) emerged as an elaboration of the software engineering construct of Actors (Jacobson et al., 1992), which, like use cases, required adaptation to better suit the needs of user interface designers.

## Roles and Personas

Both user roles and personas are effective means for capturing and conveying basic understanding about users and for informing the design process, but they differ in important ways. Personas describe users while user roles describe relationships between users and

### Use Case Essentials

Use cases, first introduced by Ivar Jacobson as the pivotal model in object-oriented software engineering (Jacobson et al., 1992), have become ubiquitous among software and Web-based applications developers but are less well known among interaction designers. A use case is simply a case of use represented in terms of the actions and variants performed by a system in interaction with external actors, which can be either users or other systems. As originally conceived, use cases were better suited for capturing systems-oriented requirements and guiding the software engineering process than for user interface design. A classic example often used to illustrate use cases is withdrawing cash from an ATM. Here is how one textbook (Kruchten, 1999) represents this task:

**Withdraw Money**

The use case begins when the client inserts an ATM card. The system reads and validates the information on the card.

1. System prompts for PIN. The client enters PIN. The system validates the PIN.
2. System asks which operation the client wishes to perform. Client selects “Cash withdrawal.”
3. System requests amounts [sic]. Client enters amount.
4. System requests type. Client selects account type (checking, savings, credit).
5. The system communicates with the ATM network to validate account ID, PIN, and availability of the amount requested.
6. The system asks the client whether he or she wants a receipt. This step is performed only if there is paper left to print the receipt.
7. System asks the client to withdraw the card. Client withdraws card. (This is a security measure to ensure that Clients do not leave their cards in the machine.)
8. System dispenses the requested amount of cash.
9. System prints receipt.
10. The use case ends.

As in this example, such conventional use cases contain many built-in assumptions about the design and implementation of the user interface. As such, they do not facilitate separating the essence of the user’s task from issues of user interface design. Moreover, user interests can be lost among the systems-oriented details, as can be seen in this use case, where the user never actually takes the money withdrawn.

**Use Case Essentials continued**…

Lucy Lockwood and I developed essential use cases in 1993 as a technique better suited to driving visual and interaction design (Constantine, 1994; Constantine, 1995). We drew on the concept of essential models introduced in a classic text on systems analysis (McMenamin and Palmer, 1984). An essential model is one expressed in abstract, simplified, and generalized form independent of explicit or implied assumptions about technology or implementation. Instead of user actions and system responses, essential use cases represent user intentions and system responsibilities. The same task of withdrawing cash when reduced to its essential form might look like this:

withdrawing cash from my account via ATM

| USER INTENTIONS | SYSTEM RESPONSIBILITIES |
|---|---|
| | 1. request identification |
| 2. provide identification | 3. verify identification |
| | 4. offer choices |
| 5. choose | 6. give cash |
| 7. take cash | |

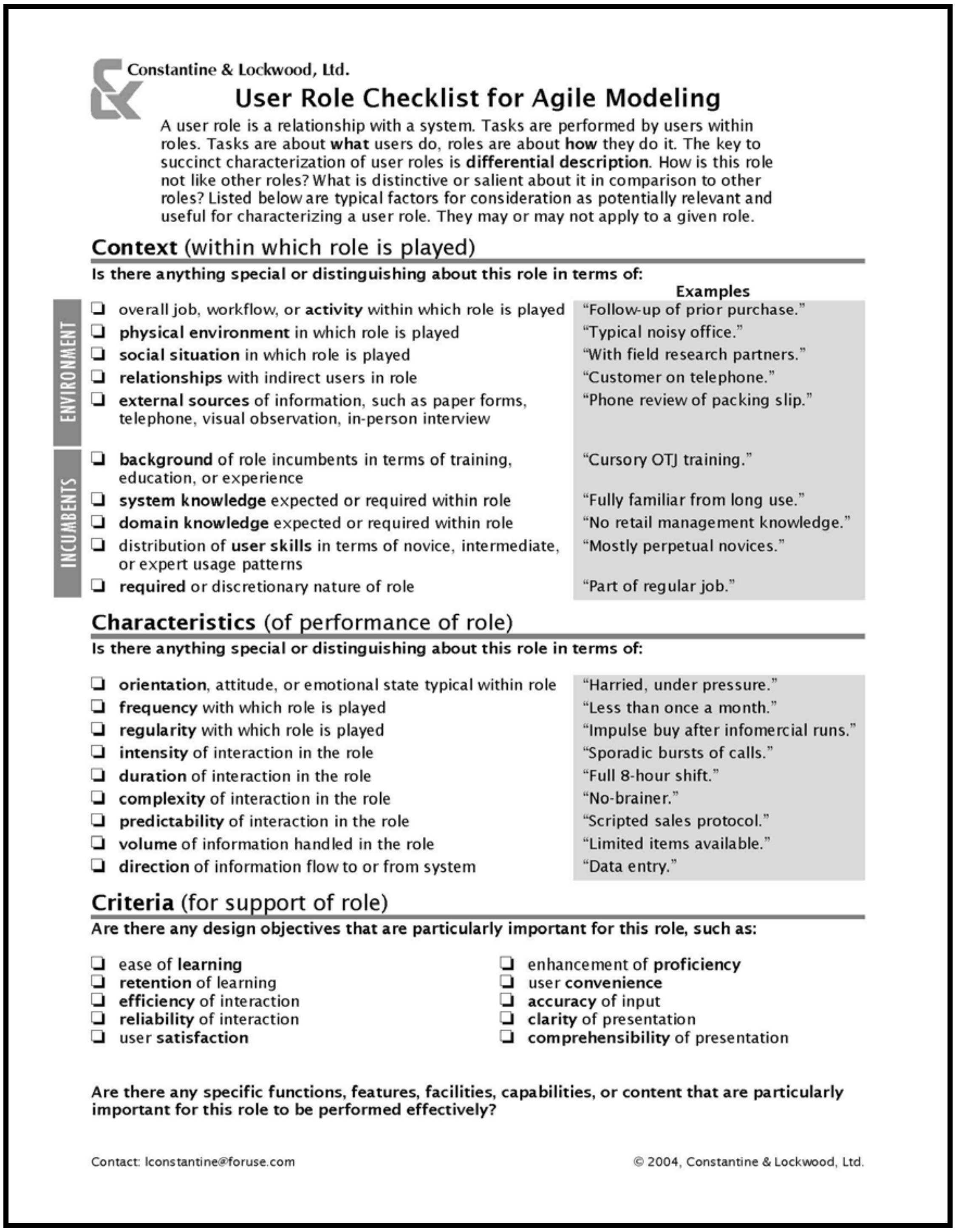
Abstraction and independence of implementation enables the modeler to focus on the essence of the task from the perspective of the user without complicating the picture by jumping ahead into details of the visual and interaction design. In this example, the abstract essence of my task as a user is just to tell the system who I am, make a choice, and get my cash. Essential models challenge the designer both to understand the true nature of the problem from a user perspective and to provide a much simplified solution. Abstraction highlights opportunities for further exploration and for innovation, such as making the choices offered reflect user intentions, such as withdrawing the usual amount, rather than bank policies and systems structures. If I have only one account and always withdraw $240 dollars from it, I should not have to tell the ATM which account and re-enter the amount on every withdrawal, for example.

Task models in the form of essential use cases have shown that they can help designers identify innovative solutions that better fit with genuine user needs (Constantine and Lockwood, 1999; 2002; Strope, 2003; Windl, 2002b).

systems. Personas are figurative models rather than abstract models, that is, they are constructed to resemble real users, even down to photos, background information, and personal history. Verisimilitude most likely contributes to the popularity of personas. They sound like people you could know, and over the course of a project can take on a reality that encourages empathy and facilitates thinking from the user perspective. What is more, many people find that the creative process of constructing personas to be engaging and energizing. Personas are fun.

By contrast, user roles do not resemble real people nor are they intended to; roles are spartan abstractions narrowly focused on those aspects of the relationship most likely to be relevant to presentation and interaction design. Compared to personas, user roles are a more technical and formally structured model.

In the broadest sense, a user role has been defined as a collection of characteristic needs, interests, expectations, and behaviors in relation to a particular system (Wirfs-Brock, 1993). In its most compact form, the form now most commonly used, a user role is represented by its Context, Characteristics, and Criteria, that is, the context in which it is played, the characteristics of its performance, and the criteria that must be met by the design to support successful performance of the role. Context includes the overall responsibilities of the role and the larger work and environmental context within which it is played. Characteristics refer to typical patterns of interaction, behaviors, and attitudes within the role. Criteria include any special functional facilities needed for effective support of the role along with design objectives of particular importance to that role. Such criteria for effective support of a role are sometimes referred to as usability or user experience attributes. Checklists and templates have been developed from extensive experience to help designers think about the central issues and judge what is likely to be most relevant for user interface design. One example is shown in Figure 2; others have been published elsewhere (Constantine and Lockwood, 1999; also www.foruse.com/publications/templates/.)



Constantine & Lockwood, Ltd.

**User Role Checklist for Agile Modeling**

A user role is a relationship with a system. Tasks are performed by users within roles. Tasks are about **what** users do, roles are about **how** they do it. The key to succinct characterization of user roles is **differential description**. How is this role not like other roles? What is distinctive or salient about it in comparison to other roles? Listed below are typical factors for consideration as potentially relevant and useful for characterizing a user role. They may or may not apply to a given role.

**Context** (within which role is played)

**Is there anything special or distinguishing about this role in terms of:**

| | | Examples |
|---|---|---|
| ENVIRONMENT | ❑ overall job, workflow, or **activity** within which role is played | "Follow-up of prior purchase." |
| | ❑ **physical environment** in which role is played | "Typical noisy office." |
| | ❑ **social situation** in which role is played | "With field research partners." |
| | ❑ **relationships** with indirect users in role | "Customer on telephone." |
| | ❑ **external sources** of information, such as paper forms, telephone, visual observation, in-person interview | "Phone review of packing slip." |
| INCUMBENTS | ❑ **background** of role incumbents in terms of training, education, or experience | "Cursory OTJ training." |
| | ❑ **system knowledge** expected or required within role | "Fully familiar from long use." |
| | ❑ **domain knowledge** expected or required within role | "No retail management knowledge." |
| | ❑ distribution of **user skills** in terms of novice, intermediate, or expert usage patterns | "Mostly perpetual novices." |
| | ❑ **required** or discretionary nature of role | "Part of regular job." |

**Characteristics** (of performance of role)

**Is there anything special or distinguishing about this role in terms of:**

| | |
|---|---|
| ❑ **orientation**, attitude, or emotional state typical within role | "Harried, under pressure." |
| ❑ **frequency** with which role is played | "Less than once a month." |
| ❑ **regularity** with which role is played | "Impulse buy after infomercial runs." |
| ❑ **intensity** of interaction in the role | "Sporadic bursts of calls." |
| ❑ **duration** of interaction in the role | "Full 8-hour shift." |
| ❑ **complexity** of interaction in the role | "No-brainer." |
| ❑ **predictability** of interaction in the role | "Scripted sales protocol." |
| ❑ **volume** of information handled in the role | "Limited items available." |
| ❑ **direction** of information flow to or from system | "Data entry." |

**Criteria** (for support of role)

**Are there any design objectives that are particularly important for this role, such as:**

| | |
|---|---|
| ❑ ease of **learning** | ❑ enhancement of **proficiency** |
| ❑ **retention** of learning | ❑ user **convenience** |
| ❑ **efficiency** of interaction | ❑ **accuracy** of input |
| ❑ **reliability** of interaction | ❑ **clarity** of presentation |
| ❑ user **satisfaction** | ❑ **comprehensibility** of presentation |

**Are there any specific functions, features, facilities, capabilities, or content that are particularly important for this role to be performed effectively?**

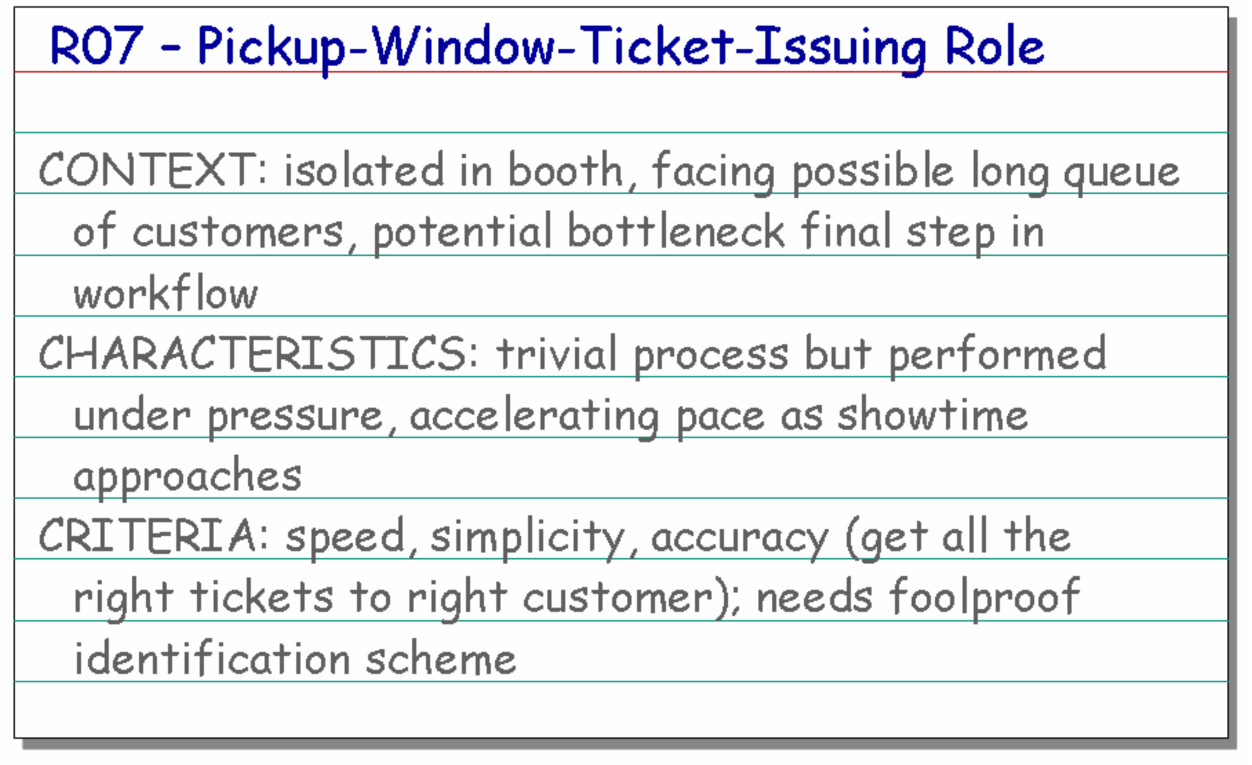
Contact: lconstantine@foruse.com

© 2004, Constantine & Lockwood, Ltd.

Figure 2 - Checklist for user role context, characteristics, and criteria.

The most popular form of user role modeling is the card-based technique employed in agile usage-centered design (Constantine and Lockwood, 2002; Constantine, 2004b). In this variant of usage-centered design, simplified models are constructed rapidly using ordinary index cards. The aim is to obtain a compact, easily created and managed model as quickly and painlessly as possible. The streamlined nature of the process makes it particularly suitable for software developed on tight schedules or using rapid iterative development, such as, extreme programming (Jeffries, Anderson, and Hendrickson, 2001), but the techniques are equally effective in more formal engineering processes and have been employed with equal success even on very large and complex projects (Strope, 2003; Windl, 2002b).

An example of a user role card is shown in Figure 3, which describes the Pickup-Window-Ticket-Issuing Role, one of a number of roles that might be played by ticket-window agents using ticketing software for a multi-venue arts center. As in this example, user roles are given names that highlight the core functional responsibilities of the role. A permanent identifier serves as a handle to facilitate filing and tracking in large projects, particularly ones in which tracing of requirements through the process may be required.


R07 - Pickup-Window-Ticket-Issuing Role

CONTEXT: isolated in booth, facing possible long queue of customers, potential bottleneck final step in workflow

CHARACTERISTICS: trivial process but performed under pressure, accelerating pace as showtime approaches

CRITERIA: speed, simplicity, accuracy (get all the right tickets to right customer); needs foolproof identification scheme

Figure 3 - Example of condensed, card-based model of a user role..

Each of the two views of users—roles and personas—has advantages and disadvantages. The very realism of personas that makes them so natural, appealing, and memorable means that the narrative can become complicated by potentially distracting details. In a fully-fleshed persona, it can be difficult for the casual reader or even the well-informed designer to know what matters and what does not, what is important for user interface design and what is not, what is an accurate reflection of real user characteristics and what is mere concoction, particular as creative invention is encouraged in constructing personas. Although the proponents of personas would argue that properly constructed personas avoid distracting information when described with “just the right detail” suitably anchored in conscientiously collected and cataloged factoids, it might be hard to rationalize the design relevance of such gratuitous details as that 9-year-old Tanner, an example persona introduced in Chapter 4, “rides his skateboard and bike, plays in the yard and nearby creek,” “arrives home at 3:15,” and “gives [his mother] a phone call” to reassure her.

Personas, because they are figurative models cast in concrete terms, can also easily cross the boundaries from user description into user interface design, thus subtly steering aspects of the user interface. Ogden, the “occasional user” of Chapter 4, is described as “wanting inline instructions, ToolTips,” and “pre-populated fields with past values,” all of which might ultimately be good ideas but are clearly descriptions of specific solutions rather than of general features of a user perspective.

In contrast with persona construction, the goal in user role modeling is to capture what is most salient in a most concise form. Differential description, in which each role is characterized primarily by those things that distinguish it from other roles, can also help to condense the information. Indeed, one of the operating principles of card-based modeling is that anything that does not fit on a single index card is too complicated and should be further simplified and condensed. In the interest of brevity, no attempt is made to create an interesting, engaging, or recognizable portrait of the user. The abbreviated description defining a user role is closest to the concepts of a skeleton or a resume-style foundation document introduced in Chapter 4, but with the added twist of structuring and focusing the narrative in predefined categories deemed most likely to be of direct value in steering the design.

Portraying archetypal users as fully formed people may have a certain humanistic appeal, but users as people are complex and multifaceted, as is reflected in the elaborations needed to lend realism to personas. The relationships users have with any particular system are, by contrast, necessarily simpler and more limited. By focusing narrowly on the relationships between users and systems rather than on users more broadly, and by employing abstraction rather than elaboration, user role modeling offers designers a more precise model of users targeted more specifically to design needs.

Another difference in the two approaches is that, with personas, the ideal is to fully describe a small number of archetypal users. User role modeling seeks to cover the playing field with a collection of interrelated but distinct descriptions of all the various roles that users can play in relation to the system. In most cases, then, user roles will outnumber personas but the personas will be more elaborate. Whereas descriptions of fully developed personas can often fill several pages, user roles are usually described on a single index card and seldom if ever take up more than a page.

## Modeling Users with Roles

In our experience, an extra step or two that helps to reveal the broad territory or sharpen details can actually speed up the modeling process. We like to begin user modeling by mapping out all the players in the story, even those who do not figure directly into shaping the user interface design. For this reason, we typically start with neither Roles nor Personas, but with Actors, the original software engineering concept that has become part of UML, the Unified Modeling Language (Fowler and Scott, 1997) widely used in software engineering. In UML, Actors are anything that interacts with a system.

To map out the complete context of use for a particular system and to define the requirements, you need to know all the Actors, whether or not these are people. Next, you need to distinguish User Actors (people) from System Actors—non-human systems that interact with the system. Because System Actors interact through other interfaces, they are not, strictly speaking, part of the user interface design problem, though they are part of the problem to be solved. System Actors imply requirements that shape internal and back-end design and become input to the software engineering and programming processes.

Among User Actors, you need to be careful to distinguish the Direct User Actors who have hands-on interaction with the proposed system from Indirect (or Mediated) User Actors who do not directly interact with a system but rather "use" it through intermediaries. These latter "off-stage" players are involved in the “story” of the system in use, but, like extras in a movie, are really part of the context within which the interaction takes place. Often they are important stakeholders, so we do not want to forget them, but we do not normally need to model them in detail for effective usage-centered design. For example, in the ticket sales application introduced earlier, the telephone ticket agents are among the Direct User Actors, whereas the customers on the other end of the phone are Indirect User Actors whose participation is mediated by the telephone staff. We design the screens for the telephone ticket agent, the on-stage Actor, but we must take into account the off-stage voice of the customer who is talking on the other end of the line. Such customers may not see the screen or touch the keyboard, but their presence as part of the Context of the Telephone-Selling Role can affect our design decisions. We may, for example, avoid audible prompts from the sales application to keep from distracting the customer or ticket agent or interrupting the conversation taking place over the telephone.

Exactly as on the stage and screen, Actors can play many different roles, so we need to identify all the Roles that all the Direct User Actors can play and within which they will interact with the system. A User Role, remember, is a relationship. Because the Role focuses on the relationship of users with a system, it captures most closely what it is important for the designer to understand to develop the most effective visual and interaction design. Telephone ticket agents might have very different relationships to the sales support system depending on whether they are actually selling tickets or responding to telephone inquiries; we might call these the Telephone-Query-Handling Role and the Telephone-Selling-Role. The Telephone-Query-Handling Role, in turn, might be played by either regular telephone agents or by a supervisor.

Of course, Actors can switch roles, so the agent on the telephone might switch among different relationships with the system even in the course of a single telephone call. An inquiry can turn into a sale or a customer about to finalize a purchase might ask for more information. The multiple roles played by a particular Actor help guide the design in terms of the flexibility and ease of navigation required to be built into the user interface.

The entire usage context can be summed up with a simple diagram called a context map, such as the one shown in Figure 4. A rectangle represents the boundary of the system being designed. System actors are shown as connected directly to that system boundary, as indeed they are. Direct Actors—the on-stage players—are shown as connected to the system by way of the various Roles they play. Indirect Actors—the off-stage players—are shown as connected to the Direct Actors that serve as intermediaries and through which Indirect Actors are involved with the system.

For a given application, the roles in a complete catalog of user roles are typically not independent of each other. Some roles, for example, are best thought of as specialized variants of others, and some may be combinations that include two or more other roles. This interdependence makes it possible to map out all the roles without a lot of repetition. For the ticketing application, the Supervising-Telephone-Sales Role, for instance, might be described as a specialized version of a Telephone-Selling Role, characterized by additional assumptions about the incumbent, the user playing the role, and special needs in terms of functional support. Whereas the Telephone-Selling Role may require only moderate training and little domain knowledge regarding the business of an arts performance center, the Supervising-Telephone-Sales Role assumes greater experience and significant domain knowledge. The Supervising-Telephone-Sales Role adds to the Criteria the need for support of special supervisory functions, such as overriding the conditions of sale or releasing seats held for VIPs.

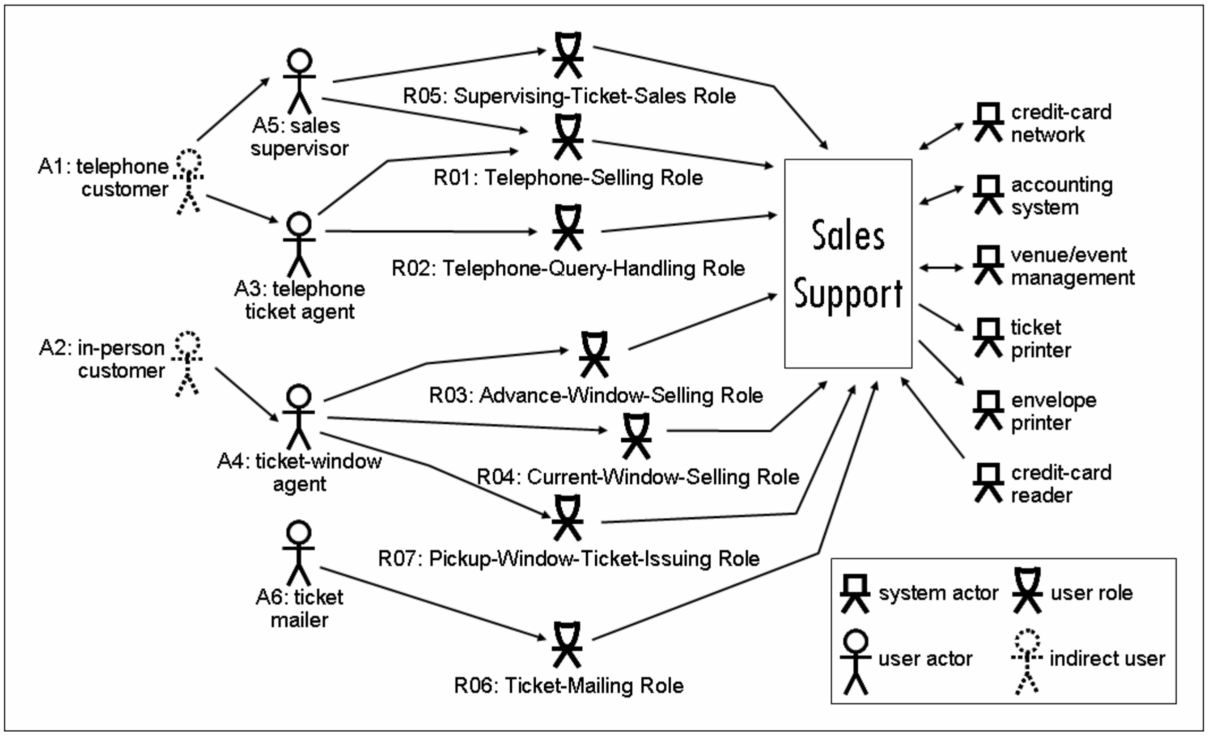


Figure 4 - Context map for the example ticketing application.

For complex problems, the relationships among user roles can be represented diagrammatically in a separate user role map, such as the one illustrated in Figure 5. In addition to specialization and inclusion, user roles may have a workflow relationship in which one role depends on the prior performance of another. For instance, the Pickup-Window-Ticket-Issuing Role in the current example is said to be preceded by the Telephone-Selling Role, as shown in Figure 5.

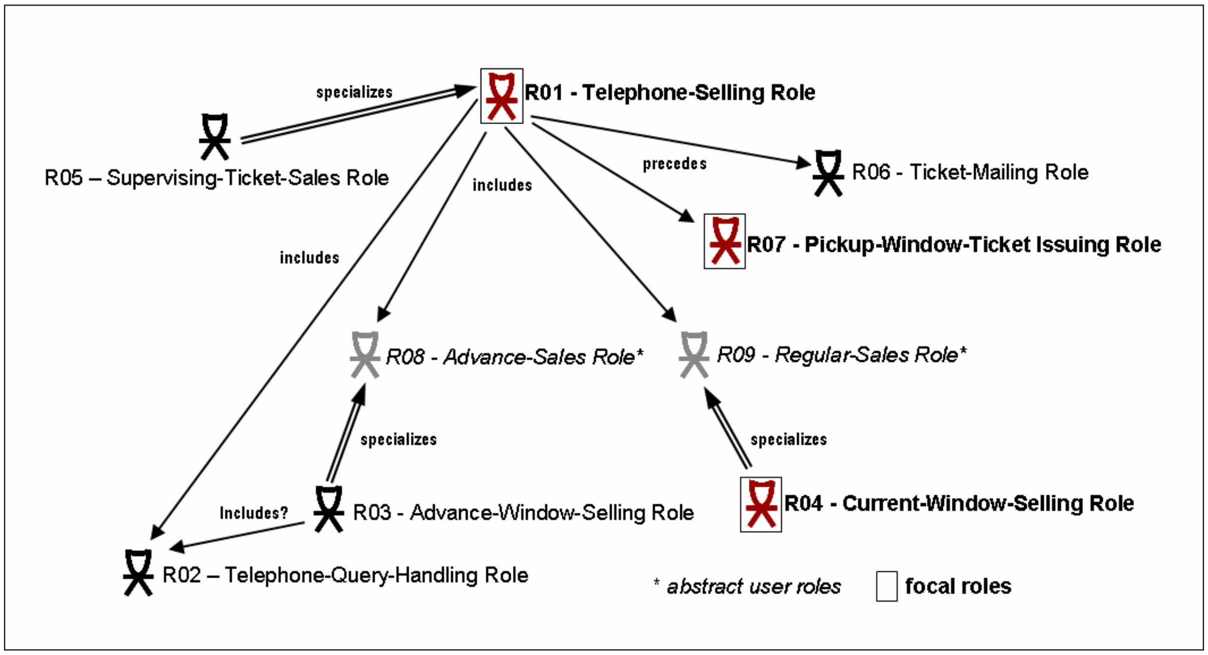

specializes
R01 - Telephone-Selling Role
R05 – Supervising-Ticket-Sales Role
includes
precedes
R06 - Ticket-Mailing Role
R07 - Pickup-Window-Ticket Issuing Role
includes
R08 - Advance-Sales Role*
R09 - Regular-Sales Role*
specializes
specializes
Includes?
R03 - Advance-Window-Selling Role
R04 - Current-Window-Selling Role
R02 – Telephone-Query-Handling Role
* abstract user roles
focal roles

Figure 5 - User role map for the example ticketing application.

Although it is useful to understand all the roles users can play in relation to a system, not all roles are regarded as equally important. Some roles will be played more frequently than others and will account for a greater portion of the use of the system. Some roles may be more important than others for the practical success of a system. For example, in the ticket sales application, telephone selling might happen every day whereas ticket windows might be staffed only at performance times over the weekend. Nevertheless, the Current-Window-Selling Role, which is responsible for in-person sales of tickets for today's performances, is very important for the business success of the ticketing application.

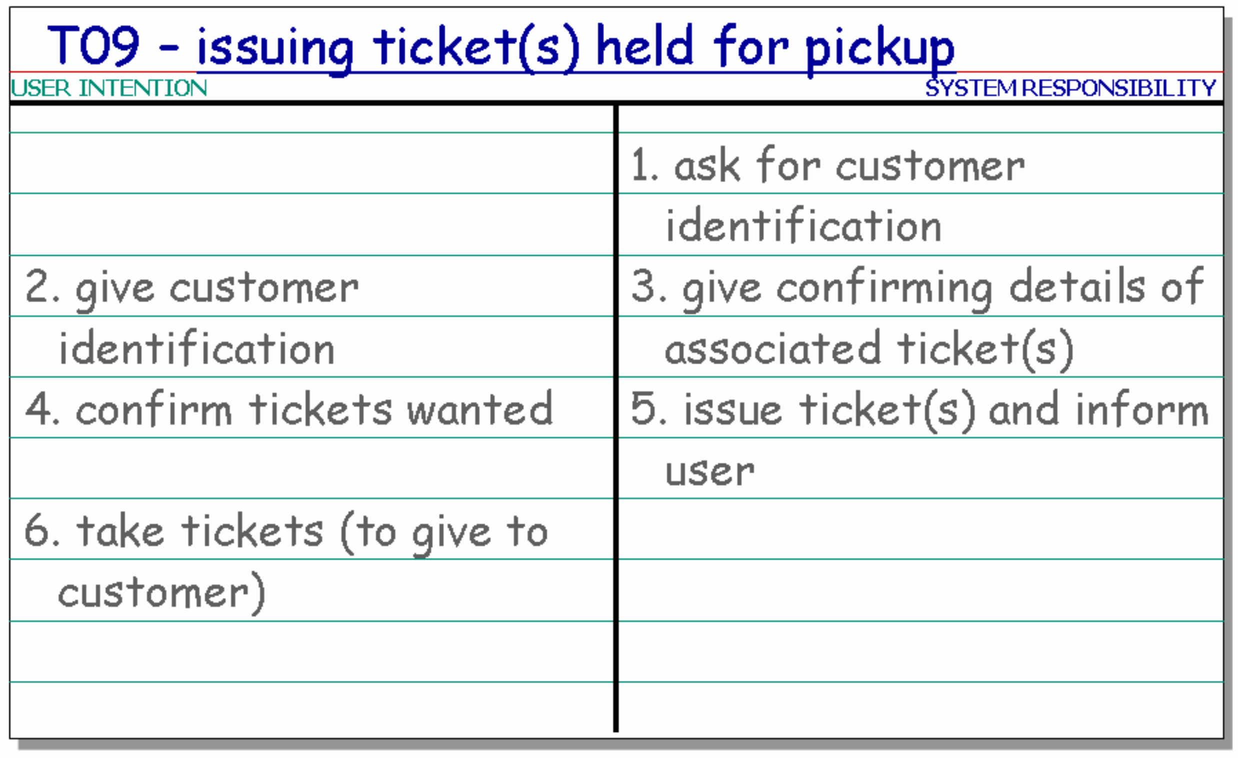
In usage-centered design, user roles are usually ranked by anticipated frequency and by business importance. The agile modeling technique for making these rankings is to sort index cards representing the roles into order (Constantine and Lockwood, 2002). On the basis of the combination of these two potentially different views of priority, a small subset of roles is distinguished as "focal" roles. Focal roles serve as a central focus for the rest of the design process, but not to the exclusion of other user roles. For the ticket selling application, the card-sorting exercise might select the Telephone-Selling Role, Current-Window-Selling Role, and the Pickup-Window-Ticket-Issuing Role as focal roles. Focal roles are like primary personas in that they are recognized as particularly important for a successful design.

## Modeling User Tasks

Ultimately, the user interface must be designed to enable users to do things, which requires more than just an understanding of users or the roles they play, but also a thorough understanding of the tasks that users must be able to accomplish in performing those roles. Indeed, task modeling is the very core of usage-centered design with its focus on user performance. Tasks are modeled with task cases, a form of the use cases introduced in object-oriented design (Jacobson et al., 1992). A task case represents a single, discrete intention carried out by a user in some role. Task cases are also called essential use cases because they are stripped down to the barest essentials of the task. Each task case is expressed as a dialog in which user intentions and system responsibilities are abstract, simplified, and stripped of all assumptions about technology or implementation. This form of description is intended to get closer to the essence of what the task is really about from the perspective of the user in a role and to avoid making unintended or premature assumptions about the form of the user interface to be designed.

For example, in support of the Pickup-Window-Ticket-Issuing Role, the task case <u>issuing held ticket(s) for event</u> might be defined as shown in Figure 6. Several things are worth noting about this example and task cases in general. Unlike use cases, which are constrained always to begin with a user action, task cases can begin with a system responsibility. As shown in this example, task cases are simplified in part by focusing on the "happy case" (Cockburn, 2001) or the normal course of interaction where everything goes well: the identification is valid, the desired tickets are found, and so forth. The rationale for temporarily hiding or ignoring the alternatives and exceptions is to promote an interface that is organized around the primary purposes as viewed from the user in role more than around the numerous less likely alternatives and exceptions that tend to dominate in the minds of programmers.

In contrast to scenarios, which are used in goal-directed design and some other design approaches, task cases represent small pieces of the performance of a user role rather than a relatively large story that has been elaborated with extra details to make it seem real and believable. Instead, as with user roles, the language is spare and abstract. This leaves open many alternative solutions for the designer to choose among. The same task case can apply whether the ticket issuer depends on credit-card verification or types the user name or scans a barcode imprinted confirmation form.



T09 - issuing ticket(s) held for pickup

| USER INTENTION | SYSTEM RESPONSIBILITY |
|---|---|
| | 1. ask for customer identification |
| 2. give customer identification | 3. give confirming details of associated ticket(s) |
| 4. confirm tickets wanted | 5. issue ticket(s) and inform user |
| 6. take tickets (to give to customer) | |

Figure 6 - Example of a task case expressed in essential form.

Task cases are identified based on the user role model in conjunction with whatever else is known about the system being designed. Many task cases follow more or less directly from the definition of roles. Among the defining criteria for the Supervising-Ticket-Sales Role referred to earlier is the need to be able to override the normal selling price or to sell specially reserved seats, which leads to formulating the task cases selling tickets(s) at discounted/special price and selling ticket(s) for special seat(s). Some task cases are implied by the purpose and overall responsibilities of a role. Successful performance of the Telephone-Selling Role requires selling ticket(s) for seat(s) for performance(s); the Telephone-Query-Handling Role requires such task cases as reviewing program/details for an event/performance and selecting event(s)/performance(s) of possible interest. The Pickup-Window-Ticket Issuing Role obviously requires issuing ticket(s) held for pickup. Other task cases may be based on requirements that are not necessarily reflected directly in the user role model. For example, requirements specifications or other artifacts might have to be reviewed to identify the need for releasing donated seats for resale.

The complete task model is expressed by the narrative bodies of all the individual task cases along with a task case map similar to a user role map but showing the interrelationships among task cases. The task case model can be checked against the user role model to verify that all user roles can be fully performed with the identified task cases and that every task case is genuinely needed for the performance of some one or more roles. The objective is to be truly comprehensive, to cover all tasks needed to fully perform all identified roles. In principle it might seem that a complete task model is an unobtainable ideal, but in practice we have found that, with well timed and thoughtful reviews along the way, it is rare to discover missing task needs late in the game. This has held true even for very large and complex applications, such as the Siemens STEP 7 Lite system (Windl, 2002b), a specialized integrated development environment for automation programming that is roughly as complex as Visual Basic. In support of 20 user roles, 270 task cases were identified, which accounted for all of the functionality needed within the system.

## From Abstract Tasks to Concrete Interfaces

Many designers develop paper prototypes or page mockups based directly on what they understand from scenarios or other task-oriented models. In usage-centered design content models or abstract prototypes serve as a bridge between task models and user interface designs. I am using the term user interface design here to cover the design of every aspect of the interface that mediates between users and systems. This includes both the visual or presentation design as well as interaction design, by which I mean the specification of the behavior of the user interface and the means by which users interact with it.

Abstract prototypes can be distinguished from the figurative prototypes used by most designers. The latter are typically expressed as mockups or paper prototypes that are intended to look like or more or less resemble real user interfaces, even if only as rough sketches. Abstract prototypes are not intended to look like the real thing, but instead embody the function and organization of user interfaces divorced from details of appearance and behavior. In this way the complex and somewhat mysterious process of designing a user interface to fit the task needs of users can be broken down into two simpler and better understood steps. The contents and organization of the user interface can be derived more or less directly from a well formed task case model, and a good visual and interaction design can, in turn, be developed straightforwardly based on the content and organization expressed in an abstract prototype, particularly one expressed in standard form using canonical abstract components (Constantine, 2003; Constantine et al., 2001). Canonical abstract prototypes are a highly structured and standardized form of wire-frame schematic that enable designers to resolve problems in layout and organization apart from details of presentation and interaction design. They use a canonical or standardized collection of abstract user interface components to model the abstract functions of the user interface as viewed from the perspective of the user. An example of a draft abstract prototype for one portion of the telephone ticketing application is shown in Figure 7. For more information about abstract prototyping and canonical abstract prototypes, see the references cited.

Elements of the user role model enter again in developing the final user interface design. The overall structure of a well designed user interface that fits closely with user needs can be thought of as an elaboration of the structure of the user role model and the structure of the task model that supports it. User roles provide direct architectural guidance such as rarely can be derived from personas, since all the parts of the interface that provide functions or information in support of a single role must form a closely interconnected set. For efficient user performance, roles that can be played by the same actor who can switch between them must be supported by parts of the user interface within easy reach of each other. Similarly, closely connected task cases must be supported together on the user interface. Conversely, roles that have little or no connections or are played by different actors can be supported through more or less distinct and separated parts of the user interface.

Although such rules may seem rather obvious, in very large, complex systems, the architectural guidance provided by the user role model and the task case map are invaluable and can help designers avoid subtle mistakes that could otherwise lead to expensive changes in organization late in development. Because the user role model is constructed to cover all supported roles played by users, and the task case model is constructed to cover all tasks needed to support those roles, it is easier to assert that a design is complete than where the design is based on a handful of personas or a few scenarios that are representative but not exhaustive. The user role and task models are easily validated through review by users, stakeholders, and independent auditors and can be cross checked using techniques like the role support matrix (Constantine, 2001). Again, even on very large systems, missing features or functions are rare with usage-centered designs based on full role and task modeling.

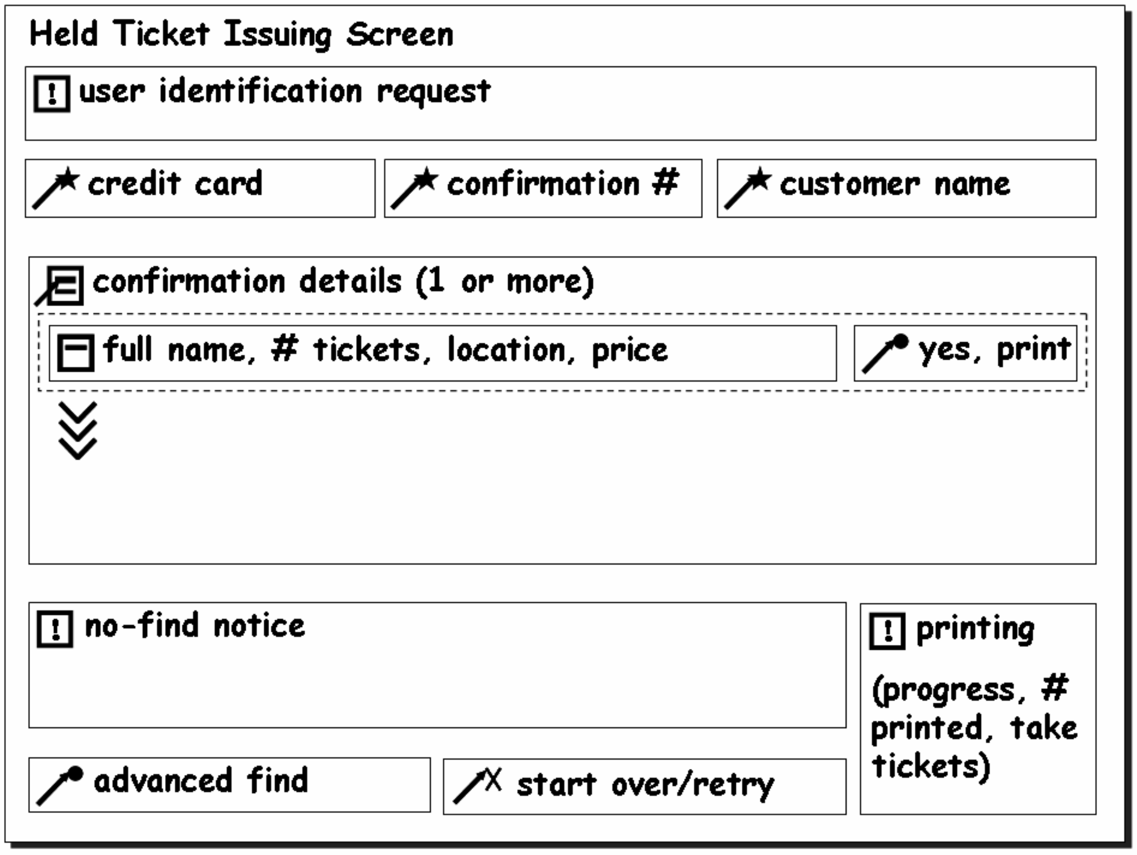

Held Ticket Issuing Screen
user identification request
credit card
confirmation #
customer name
confirmation details (1 or more)
full name, # tickets, location, price
yes, print
no-find notice
printing
(progress, # printed, take tickets)
advanced find
start over/retry

Figure 7 - Abstract prototype for ticket-issuing user interface using canonical abstract components.

Details of the visual and interaction design, too, are shaped by insight captured in the user role model. For example, returning to the Pickup-Window-Ticket-Issuing Role, the description of this role favors an absolutely mechanical, foolproof design in which the normal flow of interaction is trivial and the exceptions are all covered and handled in the most expeditious manner. If, for example, the customer is identified by name and the system finds more than one match, the screen can prompt the user to request a first name or ask whether the customer knows the price or seat section or whatever bit of information distinguishes the ticket instances found by the system. If the system is unable to find the tickets or there is some other problem, the user can be prompted to say, “I’m sorry, but the system is having some problem with your tickets. If you could step over to the Customer Service Desk to your right, someone there will be able to help you.” In this way the “possible long queue of customers” referred to in the context of the role is not exacerbated.

## Both/And Modeling

It is possible to have it both ways, to get the precision and parsimony offered by user roles along with the richness and realism of personas by combining them in the right way. In our experience, reinforced by that of numerous clients and the even more numerous practitioners we have trained, the disciplined abstraction of user roles make them better as the primary model for driving usage-centered design. Once a user role model covering the full range of user roles is available, it is possible to develop personas corresponding to selected roles, particularly the focal roles, that is, those that are the most common and most important for the success of the design.

As mentioned before, to identify focal roles, user roles are ranked in terms of the anticipated frequency with which they will be played and their relative importance to the business and practical success of the system or application being designed. In a sense, focal roles are like lead roles that take center stage in driving the design to a conclusion, while supporting roles have significant but less central influence.

At this point is where personas enter the scene. For the focal user roles, one or more personas might be constructed. By waiting until all the Roles have been catalogued and the most frequent and important ones singled out for special attention, the designer can be more confident about what user(s) to represent with personas. Personas enable us to capture the essence of the most important user roles by fleshing out roles with those elaborations that lend verisimilitude to the description. Because personas seem more "real" and are typically more recognizable than user roles, they can sometimes be more effective for engaging and drawing the attention of designers and developers.

Personas can thus become the personification of roles. One might say that a persona has character, while a role has characteristics. Characteristics focus designer attention, but character promotes identification. For the ticketing application, we might construct a couple of personas, one representing the archetypal user taking on both the Telephone-Selling and Telephone-Query-Handling Roles and one for the Pickup-Window-Ticket-Issuing Role.

In the final analysis, however, interaction design is about the relationship between a user and a system, which is precisely what user roles emphasize. Although other representations or user models may imply or include aspects of it, the user role model captures and features this relationship front and center in terms that are most relevant to user interface and interaction design. The combination of user roles with personas can offer designers a powerful modeling approach that is both engaging and precisely focused.

## In Practice

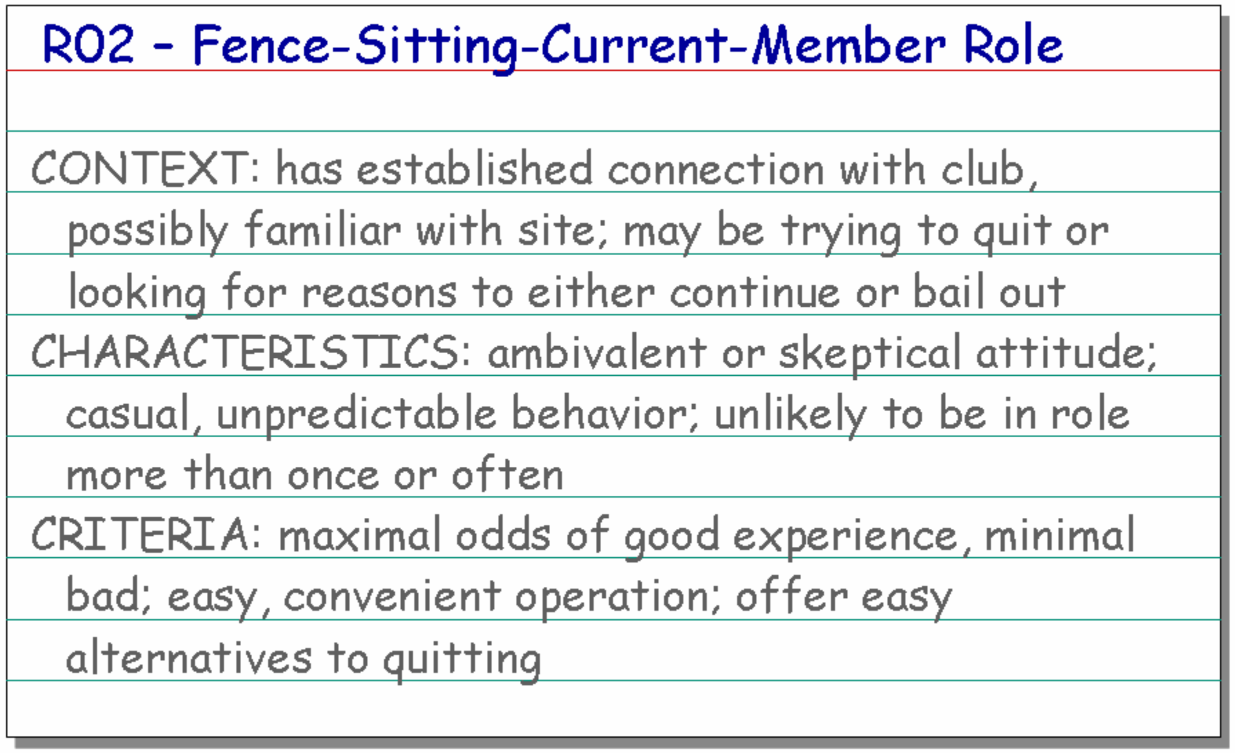
User role modeling played a significant part in one Web design project for a network of specialty book clubs. After conducting an expert usability evaluation of the existing Web site, Constantine & Lockwood, Ltd., was retained to do a complete usage-centered redesign of the combined portal and individual club sites.

The project was kicked off with a half-day collaborative modeling workshop involving 13 participants, including developers, managers, marketing staff, and user representatives. The group began by building consensus on a vision and business priorities for the project. A list of user roles was brainstormed and the 19 user roles identified were each briefly described. The roles were then individually ranked by participants on expected frequency and business importance, and the combined rankings were used to select six roles as focal, listed here starting with the highest ranked:

1. Confirmed-Current-Member Role
2. Fence-Sitting-Current-Member Role
3. Potential-New-Member Role
4. Deep-and-Narrow-Information Seeker Role
5. Specific-Information-Seeker Role
6. Former-Member Role

The prioritization and discussion of user roles yielded some unanticipated conclusions and useful insights. For example, although public relations and dealings with the media were initially considered in forming the business vision for the new site, in the context of careful comparison of frequency and business importance in relation to other roles, non-customer roles dropped to the bottom of the list. Such modeling decisions about user roles ultimately translated directly into specific design decisions, such as using easily recognized home-page links to divert non-customers, such as the press, to another site.

Another insight was the significance of former and ambivalent book club members for the business success of the remodeled site. The importance of retaining "fence-sitting" members and for winning back former members emerged as particularly important, suggesting the possibility of providing simple ways for former members to renew membership and making it easier for marginal participants to retain membership with minimal hassle. The importance of engaging both new and former members led to a navigation architecture in which the visitor could join a club from anywhere in the site, not just from the home page or a few "gateway" pages, which was the case in the previous design and on competitor sites.



R02 - Fence-Sitting-Current-Member Role

CONTEXT: has established connection with club, possibly familiar with site; may be trying to quit or looking for reasons to either continue or bail out

CHARACTERISTICS: ambivalent or skeptical attitude; casual, unpredictable behavior; unlikely to be in role more than once or often

CRITERIA: maximal odds of good experience, minimal bad; easy, convenient operation; offer easy alternatives to quitting

Figure 8 - One user role for the book club Web site.

In Figure 8 is a reconstructed description of the Fence-Sitting-Current-Member Role. In principle, anything that supports members supports this role. One could say that a well designed site that allows members to do what they want without annoyance or inconvenience is what this role needs. However, the last line of the role description suggests there is more to the story. The design should make it possible to discontinue membership online, as this gives the company one last chance to retain the member. Online cancellation could also dissuade an ambivalent customer from simply dropping out through the expedient of refusing delivery of regular monthly selections, a costly outcome for the club. However, before the member quits, the site can offer alternatives, such as, temporary suspension of membership for a few months or switching to the so-called positive-response option. Incentives could be offered, such as, a reminder that the member would earn a free book after buying only one more selection.

Rather than create an exhaustive task model, the workshop focused initially on task cases needed for support of the six focal roles. The resulting list of 24 task cases were rank ordered independently by participants on expected frequency of occurrence, on importance to customers, and on importance for business success. The rankings were compared and combined to identify focal task cases, five of which were then defined in full by the group. These included (in order, highest priority first):

1. joining a club
2. ordering book(s)
3. browsing books by category, title,…
4. seeing book-title descriptions
5. finding book(s) by criteria of interest

Over the eight week project, content models and a navigation map for the site were developed by the designer and validated by the client. Then the visual and interaction design—in the form of annotated page mockups—was completed, inspected and refined, validated with users, and delivered to enthusiastic reception from the client. However, owing to a series of contemporaneous acquisitions and mergers in which IT functions were shifted and reorganized, the design was never implemented in its entirety, although elements and ideas from it were eventually incorporated into various Web sites that came under the newly merged corporate empire.

While personas were not used in this project, it seems clear that quite a few might be needed to cover the members of clubs for interests as diverse as mystery and crime fiction, gardening, and computers and information science. Nevertheless, it certainly would have been possible to construct personas corresponding to focal roles, such as, Confirmed-Current-Member Role or Fence-Sitting-Current-Member Role. I have little doubt that this would have appealed to the client and most likely would have helped make the users more vivid to the developers. For design purposes, however, the user roles themselves captured enough about what is essential in the relationships with the Web site to enable a credible task modeling and an informed site design.

## Personas or Not

In my own work, the choice to build personas or not is made on a project-by-project basis, but the guiding principle in usage-centered design is always to model only those things and only to the extant that demonstrable payoff justifies the added effort. In some cases, reification of abstract constructs is more than worth the effort and the complexities it introduces. For designers who are more comfortable in a world of concrete and recognizable objects than in one of abstractions, in efforts where empathetic identification with users is lacking and deemed vital, or in projects gifted with ample resources and generous timetables, there seems little reason not to augment user role models with personas. On the other hand, where time is short, resources are few, and the designers are perfectly happy with succinct abstractions, user roles are most likely more than enough.

Alan Cooper has argued that it is better to design for one user, one “real” user, than to try to be all things to all users. User role modeling, however, offers a compact way to capture all the essential variants in how various users can and will relate to a new system. The designer does not have to understand everything about every user to understand the essentials of those relationships.

## References

Ahlqvist, S. (1996) “Objectory for GUI Intensive Systems: Extension.” Kista, Sweden: Objectory Software AB.

Cockburn, A. (2001) *Writing Effective Use Cases.* Boston: Addison-Wesley.

Cohn, M. (2004) *User Stories Applied for Agile Software Development.* Boston: Addison-Wesley.

Constantine, L. L. (1994) “Essentially Speaking,” Software Development, 2 (11). Reprinted in L. L. Constantine, *The Peopleware Papers.* Upper Saddle River, NJ: Prentice Hall, 2001.

Constantine, L. L. (1995) “Essential Modeling: Use Cases for User Interfaces,” *ACM interactions 2* (2).

Constantine, L. L. (1998) “Abstract Prototyping,” *Software Development, 6* (10), October. Reprinted in S. Ambler and L. Constantine, eds., *The Unified Process Elaboration Phase.* San Francisco: CMP Books, 2000.

Constantine, L. L. (2001) “Creative Input: From Feature Fantasies to Practical Products.” In L. L. Constantine, ed. *Beyond Chaos: The Expert Edge in Managing Software Development.* Boston: Addison-Wesley.

Constantine, L. L. (2003) “Canonical abstract prototypes for abstract visual and interaction design.” In J. Jorge, N. Jardim Nunes, and J. Falcao e Cunha, Eds. *Interactive Systems: Design, Specification, and Verification.* Proceedings, 10th International Workshop, DSV-IS 2003, Funchal, Madeira Island, Portugal, 11-13 June 2003. Lecture Notes in Computer Science, Vol. 2844. ISBN: 3-540-20159-9 Springer-Verlag.

Constantine, L. L. (2004a) “Beyond User-Centered Design and User Experience.” *Cutter IT Journal, 17* (2), February.

Constantine, L. L. (2004b) *Agility and Usability.* Cutter Executive Report.

Constantine, L. L., and Lockwood, L. A. D. (1999) *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design.* Reading, MA: Addison-Wesley.

Constantine, L. L., and Lockwood, L. A. D. (2001) "Structure and Style in Use Cases for User Interfaces." In M. van Harmelan (ed.), *Object Modeling and User Interface Design.* Boston: Addison-Wesley.

Constantine, L. L., and Lockwood, L. A. D. (2002) “Usage-Centered Engineering for Web Applications.” *IEEE Software, 19* (2), March/April.

Constantine, L. L., Windl, H., Noble, J., and Lockwood, L. A. D. “From Abstraction to Realization in User Interface Design: Abstract Prototypes Based on Canonical Components.” Working Paper, The Convergence Colloquy, July 2000. www.foruse.com/articles/canonical.pdf

Cooper, A., and Reimann, R. M. (2003) *About Face 2.0: The Essentials of Interaction Design.* New York: Wiley.

Fowler, M., and Scott, K. (1997) *UML Distilled.* Reading, MA: Addison-Wesley.

Heumann, J. (2003) “Use Case Storyboards: Integrating Usability with RUP and UML.” In L. Constantine, (ed.) *Performance by Design: Proceedings, forUSE 2003, Second International Conference on Usage-Centered Design.* Rowley, MA: Ampersand Press.

Jacobson, I., Booch, G., Rumbaugh, J. (1999) ***The Unified Software Development Process.*** Reading, MA: Addison-Wesley.

Jacobson, I., Christerson, M., Jonsson, P., and Övergaard, G. (1992) *Object-Oriented Software Engineering: A Use Case Driven Approach.* Reading, MA: Addison-Wesley, 1992.

Jeffries, R., Anderson, A. Hendrickson, C. 2001 *Extreme Programming Installed.* Boston: Addison-Wesley.

Kruchten, P., Ahlqvist, S., and Bylund, S. (2001) User Interface Design in the Rational Unified Process. In M. van Harmelan, Ed., *Object Modeling and User Interface Design.* Boston: Addison Wesley.

Patton, J. (2002b) “Extreme Design: Usage-Centered Design in XP and Agile Development.” In L. Constantine (Ed.), *forUSE 2002: Proceedings of the First International Conference on Usage-Centered, Task-Centered, and Performance-Centered Design.* Rowley, MA: Ampersand Press.

Patton, J. (2003) “Improving Agility: Adding Usage-Centered Design to Agile Software Development.” In L. Constantine, (ed.) *Performance by Design: Proceedings, forUSE 2003, Second International Conference on Usage-Centered Design.* Rowley, MA: Ampersand Press.

Strope, J. (2003) “Designing for Breakthroughs in User Performance.” In L. Constantine, ed., *Performance by Design: Proceedings of forUSE 2003, the Second International Conference on Usage-Centered Design.* Rowley, MA: Ampersand Press.

Windl, H. (2002a) “Usage-Centered Exploration: Speeding the Initial Design Process.” In L. Constantine, ed., *forUSE 2002: Proceedings of the First International Conference on Usage-Centered Design.* Rowley, MA: Ampersand Press.

Windl, H. (2002b) “Designing a Winner: Creating STEP 7 lite with Usage-Centered Design.” In L. Constantine, ed., *forUSE 2002: Proceedings of the First International Conference on Usage-Centered Design.* Rowley, MA: Ampersand Press.

Wirfs-Brock, R. (1993) “Designing Scenarios: Making a Case for a Use Case Framework.” *Smalltalk Report,* November-December.