Presenter Information

Since 2000, Jeff Patton has championed the incorporation of User Centered Design thinking into traditional and Agile software development approaches. Through writing, teaching, speaking, and practice Jeff works to introduce developers, analysts, product managers, and others to simple techniques to help them incorporate UCD thinking and techniques into their day-to-day work practices. Jeff has found that incorporating user-centric thinking along with traditional requirements development and testing practices drives toward the ultimate goal of better software for end-users and more profitable software for those who pay for its design and development.

Since 2004 Jeff has been a consultant with ThoughtWorks working in a wide variety of domains from healthcare to stock portfolio management. Within TW Jeff has helped build the recognition of user-centric thinking as a critical success factor in their Agile projects. This type of thinking has been “baked in” to ThoughtWorks QuickStart Analysis approach.

To help build awareness of the subject of Agile development, User Centered Design and usability, in 2004 Jeff started the Agile-Usability discussion group hosted by Yahoo Groups. At over 1,300 members, the group keeps a running dialog on the subject, and helps provide a starting point for those interested in getting help applying UCD and usability practices on Agile projects.

Jeff is currently a columnist for StickyMinds.com and IEEE Software. He’s at work on the forthcoming book “Agile Development Outside-In” which published by Addison-Wesley in Cockburn & Highsmith’s Agile Development series.

Website including writing, past presentations, and blog: www.agileproductdesign.com
Course Information

Summary & Participant Benefits
Leveraging what you understand about your application’s users is central to a user centered design approach. User personas have been touted as solution for improving the quality of user experience. But what exactly is a persona? How do you create one? And, how does a persona differ from other popular approaches to modeling users such as actors, user roles, or user profiles?

In this tutorial you’ll how user models fit into a holistic user centered design process. You’ll learn about various types of user models. Through discussion and practice you’ll learn how to create simple role models, user profiles, and personas. You’ll learn how to leverage these user models to identify valuable features for your product, and overarching characteristics the design of your product must have to be successful for your target users. This tutorial is fast paced, information dense, and full of collaborative activities to give you practice with the concepts and techniques introduced.

Session One: Creating Simple User Models 90 minutes

Goals
- Understand the where user models fit in a user centered design approach
- Understand Actors, Roles, Profiles, and Personas as for simple categorizations of user models
- Understand how one user model can evolve into another based on the project needs

Exercises
- User role Modeling
- User role profiling

Session Two: Leveraging User Models 90 minutes

Goals:
- Understand why and how to communicate user models within your organization
- Leverage user models to identify product feature opportunities
- Leverage user models to identify necessary product design characteristics

Exercises
- Persona poster creation
- Feature and design imperative identification
Today we’ll cover these three areas

Understand:
- The user model’s place in a software development process
- How to build simple relevant user models
- How to leverage a user model to make design decisions

I hope to demystify what often seems a confusing subject in software design and development

Let’s look closer at people and why they user software
Personas, Profiles, Actors & Roles: Modeling Users for Successful Product Design

Norman's simple model for a human in pursuit of a goal

- problem or goal
  - How I'd like to feel, or what I'd like to achieve
- goal evaluation
  - is my goal met or problem resolved?
- action evaluation
  - did that action deliver the results I expected?
- task
  - take some action
- goal
  - How I'd like to feel, or what I'd like to achieve
- goal evaluation
  - is my goal met or problem resolved?
- action evaluation
  - did that action deliver the results I expected?
- tool
  - information and tools

Distilling this down to goals, tasks, and tools

- problem or goal
  - How I'd like to feel, or what I'd like to achieve
- goal evaluation
  - is my goal met or problem resolved?
- action evaluation
  - did that action deliver the results I expected?
- task
  - take some action
- goal
  - How I'd like to feel, or what I'd like to achieve
- goal evaluation
  - is my goal met or problem resolved?
- action evaluation
  - did that action deliver the results I expected?
- tool
  - information and tools

Software contains features that support a number of tasks and a number of goals

- goals
- tasks
- software features

Jeff Patton, jpatton@acm.org
www.AgileProductDesign.com
Software products support a variety of users and their goals.

goals → tasks → software features

In organizations where users are paid to use the software, user goals are driven by business goals.

goals → tasks → software features

All these goals mean lots of tasks, and lots of potential features in our software.

Having a good list of users helps us understand functional scope.

How many different types of users will use this software?

What goals will they be in pursuit of?

What tasks will they need to perform?

Which of those tasks will the software we design support?
Look closer at the people engaged in using your tool – what about them has relevance to the tool’s design?

- What do your users know about using computers? - assuming we’re building software
- What do they know about the goal they’re attempting reach? Have they done this before?
- How often do they do this?
- When and where are they when they’ll use the software you design?
- If they use other software like this - what expectations might they have about your software?

Questions like these help us understand characteristics our software should have to best serve these users.

How do we go about describing users in the most relevant way?

The humble “actor” gives a common name for a user type

In use case modeling, actors are people who interact with our system.

They’re often described using job titles or a common name for the type of user.

- accounts payable clerk
- manager
- cashier
- customer
The “role” names a relationship between a user type and a process or a software tool

A user role generally refers to a user’s responsibility when using a piece of software or participating in a business process.

- AP voucher enterer
- administrator
- on-line payment checker

Both actors and roles name a relationship with some entity

That relationship may be between a person and:
- Their organization
- A business process
- A software tool
- Any other entity

That relationship may be between a person and:
- Their organization
- A business process
- A software tool
- Any other entity

Both actors and roles name a relationship with some entity

An individual may change their role as their goal or responsibility changes.

Changing roles is like changing hats.

For our purposes, that entity is the software we intend to design.
Roles decompose into finer grain roles

Consider the conference we’re attending

Conference attendee
- Tutorial attendee
- Keynote talk attendee
- Lunch consumer
- Break food consumer
- Wireless internet user
- Hotel guest

Each conference attendee role can be decomposed to a role that more precisely describes my goal or responsibility relationship with the conference.

enough talk: Let’s practice thinking about roles

3 general roles:
- Attendee
- Speaker
- SD Staff

Better Software Development Magazine (BSD) has come to you and asked you to build them a better conference website. This website will support potential attendees of several conferences hosted by BSD. It will need to support the needs of potential attendees, speakers, and BSD employees who manage the conferences.

In a small group – 3-4 people, brainstorm a list of candidate roles for the system.

Try using the form “thing-doers” such as “website browser” or “presentation proposer.”

Remember the primary rules of brainstorming:
- No discussion – just ideas
- Quantity matters more than quality
- Keep it fun – suggest silly roles

Timebox this to 5 minutes – the team with the most roles wins.

Prioritizing user types is important

For the software tool we intend to build, which user types are the most relevant to the design?

This depends on the business case.

Why is the software product being built?

What business objectives do we hope to achieve?

Which of these user types is it most critical that we support to achieve our objectives?

Refer to these user types at primary, or focal.

For a typical system, expect 2 or 3 focal users.
Choose focal users - the users that best advance BSD’s business objectives

With your group, choose 2 or 3 focal roles for the BSD’s new website.

“We need to add features to the website to begin to build community all year round. When people come to a conference, they make valuable connections with each other, we want them to build those connections... to plan on coming next year and continue to grow relationships. The conferences drive BSD.”

--Tammy, BSD

Profile users to identify relevant characteristics about them

To help us understand the characteristics of our users that might have bearing on our design, construct a profile containing information about the type of user relevant to the software being created.

1. # of users that occupy this user type
2. General responsibilities or activities
3. Computer skills
4. Domain expertise
5. Goals: how does this software tool help this user reach their goals?
6. Pain Points: what nagging problems can this software help solve?
7. Usage Context: where will this software be used?
8. Software Ecosystem: what other software tools does this user type rely on?
9. Collaborators: who does this user work with to help reach their goals?
10. Frequency of Use: how often is this type of user likely to use this software?

Creating profiles from assumptions and facts

Quickly creating profiles from assumptions allows us to find out what we do and don’t know about our users.

There’s danger in basing critical decisions on software functionality on assumptions. But, before allocating time to research, the assumption based profile will help you estimate how much research you’ll need.

Interaction designer that create personas from assumptions refer them as an assumption-based persona, or a persona hypothesis.
Profile your users using assumptions

Choose one of your focal roles. As a group, for that role, discuss the following characteristics and record them for your focal role.

1. # of users that occupy this user type
2. General responsibilities or activities
3. Computer skills
4. Domain expertise
5. Goals: how does this software tool help this user reach their goals?
6. Pain Points: what nagging problems can this software help solve?
7. Usage Contexts: where will this software be used?
8. Software Ecosystem: what other software tools does this user type rely on?
9. Collaborators: who does this user work with to help reach their goals?
10. Frequency of Use: how often is this type of user likely to use this software?

Backfill profiles with facts

Given assumption-based profiles, you can identify the areas where your information is sparse or incomplete. You can use research to backfill your profiles with facts in critical areas.

- Interviewing users from target user groups
- Observing users
- Questionnaires
- Existing published demographics
- Existing published research
- Customer service records and representatives
- Sales and marketing
- Usability testing
- Focus groups

Interview someone nearby

Interview technique: ask your interview subject to recall a specific event and describe to the best of their recollection how that event took place.

Ask them to describe their experience reviewing the conference website before deciding to attend.

- Where were they?
- How long did it take?
- What computer equipment or software was used?
- What did they most enjoy about the experience?
- What was most annoying about the experience?
**Distill your user model to communicate information most relevant to the design of the software**

Of the assumptions and facts gathered, what are most relevant to the design of this software?

What could you remove to more concisely communicate to:

- analysts
- UI designers
- developers
- business stakeholders

---

**Personas make user data more tangible**

Profiles contain general characteristics about your groups of users.

A persona is an archetypal user that is derived from specific profile data to create a representative user.

A persona is more tangible, less ambiguous, easier to envision, easier to empathize with.

Personas with irrelevant or stereotypical information in them will damage user understanding and empathy.

---

**Characteristics of a good persona**

- Name
- A role or job title
- Quotes in the persona's language
- Relevant demographics
- Descriptions that reveal goals, motivations, pain points
- Descriptions that describe primary activities this user type will engage in.
Build a simple persona from your profile data

Include:
- Name
- A role or job title
- Quotes in the persona's language
- Relevant demographics
- Descriptions that reveal goals, motivations, pain points
- Descriptions that describe primary activities this user type will engage in.

You’ve built several types of user models... congratulate yourself.

How do we make this user model relevant?
Feature opportunities describe the good ideas the good ideas that result from thinking about your users

As you discuss, speak with, and observe your users, you’ll get great ideas for product features – features that will really help your users.

Include these feature opportunities in your profile or persona

Design imperatives describe good characteristics the software should have based on the user type

Inside your user profile are clues about the type of user interface and user interface characteristics needed by your user.

Document these as design imperatives.

Think about:
- ease of learning
- retention of learning
- efficiency of interaction
- reliability of interaction

Discuss and record feature opportunities and design imperatives

What feature opportunities are particularly valuable to this user type?

What characteristics must the design have to be suitable for this user type? (design imperatives)
User modeling distilled

1. Identify actors or roles - take your pick, or mix as you see fit
2. Prioritize based on relevance to the product's business case
3. Profile to identify details relevant to design
4. Personify to better communicate user types
5. Identify feature opportunities
6. Identify design imperatives
7. Communicate your user model with their relevant feature opportunities and design imperatives - this communicates the relevance of your user model

Why aren’t user models commonly built?

Self-referential design is the behavior we all revert to

User models help us understand why our users aren’t us. Much of software is built by organizations where employees are the target users. In situations where individuals within the company are example users, it’s tempting to avoid user modeling. This is “for-us-by-us” software.
User models are often not leveraged

Leverage your user models
- Prioritize them to help with adding or removing functionality from scope
- Identify user test subjects
- Identify alpha/beta testers
- Compare them with your eventual actual users to identify bad assumptions and new user constituencies
- Post them in the area your team works to help team members empathize with target users and make better tactical design decisions

User models create a common design target

Avoid arguments about what the users want or could best use by pointing to the user model.
Target the user model

We design and build software to create value for the business that pays for it.
Value doesn’t usually come from the delivery of the software, but from the use of the software.

Understanding users is critical to getting value out of our software.

Modeling users is a simple first step to clearly communicating our design target to everyone involved in software design and development.
Personas, Profiles, Actors & Roles: Modeling Users for Successful Product Design

Jeff Patton
jpatton@acm.org
AgileProductDesign.com
Essential User Modeling

A user model distills information about the users of your software that is relevant to the design of that software. Create a user model to help validate choices you'll make regarding features, usage characteristics, and general look-and-feel. Validate by referencing the user model when making those choices and asking "Is this a good decision for this kind of user?"

User models help winnow down your choices

We've got a lot of choices to make when creating software: What features should the software have? How will those features hang together? Should usage be fast, or informative, or both? Should look and feel be sexy and slick, simple and minimal, busy and information rich?

It's likely not you or your team who will use this software, so you're making these choices on behalf of someone else. It's likely that not just one person will be using the software, so you're making these choices on behalf of lots of someone elses. What's more, many of those someone elses may have contrasting opinions about the decisions you're making - which leaves you in the position of deciding whose preference to favor.

A user model distills information about your users relevant to all these choices into a tight, useful package - at least when done right. When faced with choices about your software, or when doing routine validation of usage, or look and feel, find the most relevant user or users in your model and ask: "Will this work for this person?"

Alfred E. Newman and Blended Self-Substitution

In the absence of a user model, most of us substitute the user we know best - ourselves. But sometimes we know a little about the user, or a variety of users - but not enough. In those cases we might blend what we know about users with our own preferences. In these cases we often end up with an odd or near-impossible hybrid: a elderly woman with the computer skills of your lead developer, or teenager with the esthetics of your CEO.

Mad Magazine, published since 1952, features on its cover a picture of the fictional character Alfred E. Newman. In a typical cover Alfred's face is blended with a topical pop culture figure usually resulting in a funny, but frightening, blended person.

When creating user models, be on the lookout for blends of what you know about your audience, and potentially out of place characteristics and preferences from others in the room.

Avoid Self-Centered Design

You might be saying to yourself "I make those sorts of decisions all the time without a user model - why would I waste time building this thing?" If you're not saying that to yourself, I suspect you know someone who will say it to you. The real truth is that these decisions are made with respect to some understanding of a target user. In the absence of a user model, most of us substitute the user we know best - ourselves.
Have you ever been in a room where two people are arguing about some aspect of the software both saying: "I think the user would prefer it this way"? In the absence of a some shared user model the user they're likely talking about is themselves. Arguments like this often go badly.

Build a user model to avoid self-referential design decisions

**Communicate and create empathy**

The process of creating software can be long and involve lots of people - lots of people that weren't in the room when choices were discussed, and decisions made with respect to our target users. People like developers, business analysts, technical writers, or testers may not understand where these decisions came from. What's more, they may have more detail level decisions to make on their own.

Creating a user model and displaying it effectively communicates to others the basis for many design decisions.

A user model displayed and referenced by others allows them to gain understanding and empathy for users and make detail decisions with those users in mind.

**Different types of user models**

Modeling users isn't a new idea. Very little in this book is. Various software analysis and design approaches use different mechanisms to model users. I'm going to generalize models into four types:

* Actor
* Role
* Profile
* Persona

### Actors

An actor refers to the common name used for a type user of the system. A typical actor name might refer to the job title of a user, or simply "customer" for user of a consumer focused product. Identify actors and annotate them with the goals this type of user would like to fulfill with the system.

Actor is a term used primarily in use case modeling. A use case models the interaction between multiple parties - or actors. An actor can refer to a person, a system, a part of a system, an organization - anything you’d like to model the behavior of using a . For our use we’ll talk about human types of actors.

Typical examples might be

* Cashier
* Customer
* Manager
* Accounts Payable Clerk
An actor typically references a common name for a type of user. Often they take the form of a job title. In most cases the name is common with respect to the organization paying for the software to be built. For instance actor names like "customer" and "manager" are relationships individuals might have with an organization. When you use an actor name within that organization, the people you’re talking with typically know who you’re referring to and will infer characteristics of that type of person from that actor name.

**Pair named actors with goals**

To make an actor or list of actors more useful, we might pair the actor name with a goal or list of goals with respect to a process or piece of software we’re considering.

For example, let’s say we were building a point of sale system. Given an actor like cashier we might list a few goals like:

- ring up sales for customers quickly
- do customer returns
- look up prices for customers
- keep the cash drawer balanced

When creating an actor or list of actors, pair them with the goals that motivate their use of the system you’re designing.

**User Roles**

A user role names a type of user engaged in reaching some goals with a system. A user may change roles as they change goals. For example upon finding something interesting in an on-line website a "casual browser" might change to an "impulsive buyer" or a "skeptical comparison shopper." Use the form "thing-doer" to help name roles. For extra credit use the "adjectived-thing-doer" form to capture important characteristics of the user in the role name.

The term "role" is used in a variety of contexts. It refers to the goal relationship someone has with tool or process. Think of a role as a hat a person puts on. Throughout the day, or even moment to moment, a person can change their role as they might change a hat. Their role might change as their goal with respect to the tool or process changes.

Typical roles might be:

- Casual browser
- Impatient buyer
- Order taker
- Time tracker
- AP voucher enterer

In these example roles names I’ve used a "thing-doer" form. The role names all refer to some type of work or activity a person is engaging in with tool or process. Someone wearing the hat of a particular role has the objective of completing that work or activity. Reading a role name tells me what the person is engaged in doing - although not necessarily why.
For "extra credit" I’ve used the decorated form of "adjectived-thing-doer" such as "Impatient buyer." The adjective tells me a bit about the state of mind of a person engaged in this activity. I can assume an impatient buyer in a store would like to find what they need quickly and leave.

The form of role I’ll gravitate to here is from Constantine & Lockwood’s Software for Use.

**A good role name reflects the goal a user has with respect to the software**

For example if I were building a kiosk in a large book store to help people find books I could define roles for the various people who’d use that kiosk. Suppose I have a type of customer who shows up at my book store in a hurry looking to find what they need and get out quickly. I might be tempted to call this kind of person an "impatient buyer." However, the software I’m building doesn’t allow it’s users to buy – only to find the location of books in my large bookstore. A better name for this role might be "impatient location searcher." Such a role name more accurately reflects the goal with respect to the system.

**A person may take on many roles**

A person changes roles as they change goals

A user may change roles as they change goals. For example upon finding something interesting in an on-line website a "casual browser" might change to an "impulsive buyer" or a "skeptical comparison shopper."

**A variety of actors my fulfill the same role**

Within an organization we might find an Accounts Payable Clerk. The clerk may report to an Accounts Payable Manager. Within that organization, those job titles may be significant. If we’re writing software to support the Accounts Payable department, that software may support the routine activity of AP voucher entry. That routine activity might be engaged in by both the AP clerk, and the AP manager. A role name like "AP voucher enterer", although it sounds a bit forced, let’s us focus on person engaged in this activity – not a job title which may have little baring on that activity.
User Profiles

Given and actor or user role we can profile that type of user by adding information about a variety of characteristics such as number of users, age ranges, computer expertise, or frequency of use. A user profile helps supply more relevant information from which to base design decisions and to better communicate that information to others who need to understand decisions made or make their own detail decisions. Build a profile from information acquired through research or captured from assumption.

An actor or role may be profiled by adding relevant information about the types of people occupying that actor or role.

Given an actor and goal or a user role we know a little about the user of our system, but likely not enough to run through a subset of design choices and make some decisions. It may help to add some details to our user such as age range, computer expertise, experience with the software they’re using or process they’re engaged in, the place they’re likely using the software... and the list of information relevant to your design could be endless.

When we start to identify characteristics of our user, we begin to build a profile that helps us better understand that user.

Consider profile characteristics relevance to your design

There are a huge number of details we might consider about any type of user. Some of those details are more relevant than others to the design of your software. For example when designing software to support a business process, a characteristic like gender is likely not relevant. I suppose there’s always an exception, but whether our user is male or female may not have much baring on their goals when using the software or how they might use it. However, if we’re designing a consumer product that caters to specific market segments, how we design the product for stay-at-home-moms might be different than how we define it for sports-fanatic-men.

Most characteristics may have some baring on design, but some have more than others. For example frequency of use is a characteristic that is very relevant to an information kiosk in a museum that’s likely used very infrequently by its target users.

For each characteristic you profile, consider it’s relevance to the design choices you need to make.

For characteristics that have big impact, make sure the information you have regarding that characteristic is comprehensive and reliable.

Consider a variety of characteristics

There’s a variety of characteristics you might consider. Here’s a list, not comprehensive, but useful as a starting point.

# of people in this user constituency
How many people occupy this user constituency? This helps us understand the possibly diversity of opinions in the role and the impact of our design choices. Do our choices impact ten people, or then thousand?

**General responsibilities or activities**

What general responsibilities or activities might this person engage in when using the software? How frequently are some activities done relative to others? This helps us understand what activities the software might support, and which of those activities are more relevant. If you started personifying a user role, you might should find the role implied some activities this person engaged in order to meet the goals indicated by the role.

**Computer skills**

How skilled is this person using computers and software in general? Knowing this helps us select specific user interactions that will be learnable and usable by this user type and what tradeoffs we can make. For example if one possible interaction is inexpensive to implement but would be hard for beginners to use, knowing the primary user profile is expert allows us to use a cheaper to implement interaction.

**Domain or subject matter experience**

How much can the software assume this user knows about their business domain or the subject matter the software deals with? Is the software responsible for "teaching" the user and can we freely sprinkle the software with jargon and concepts understood by experienced people? For example "rack rate" is a term used in the hotel industry to refer to the published full price of a hotel room. You likely won't find that jargon used prominently in your favorite travel site since the site hopefully assumes we're not hotel industry experts.

**Success goals**

How does this user evaluate success? What success goals motivate this user to use our software? If we know what our user considers as success, we can evaluate if the software is helping this user achieve success. Software that helps the profile succeed will ultimately make them happy. During interviews I'll usually ask the interviewee to "tell me about a good day." I'll usually get a feel for what constitutes success from that answer.

**Pain points**

What causes this user trouble outside the software? What common trouble does the user hope to avoid by using this software? If we know this about our user we can evaluate whether the software is helping the profile avoid or more quickly resolve issues. During interviews I'll usually ask the interviewee "tell me about a bad day." I'll usually get an idea of what's painful from that answer.

**Usage locations**

Where will the user likely use this software? Is it a single place, or a variety of places? On what kind of device – desktop computer, notebook, iPhone? What constraints does this location bring to the software? For example travelers won’t have always internet access. Users who spend a lot of time away from their computer won’t have the ability to get on-screen notifications timely.

**Software & tool ecosystem**
What other hardware or software products does this user consistently use? Knowing this allows us to leverage usage patterns they may already be familiar with. For example if I use Excel all day, if you give me an application that works much like Excel, I may learn it faster than if it were more like something I’m unfamiliar with.

**Collaborators**

Who does this user work with or speak with on a regular basis? Does this user need to collaborate with someone to achieve their goals with the software? Knowing this we can evaluate how the software could help this routine collaboration.

**Expected frequency of use**

How frequently do we think this type of user will use the software we’re considering? Given an understanding of the users’ likely frequency of use, we can determine how learnable the application needs to be. Applications used more frequently become familiar and users become proficient faster. Proficient users often need speed-keys or alternative faster ways to perform frequent and familiar tasks. Alternatively users with infrequent use may be constantly re-learning the application. This means the software should be learnable and certainly not rely on training or memorization.

**Profile characteristics come from research**

If you’ve read this incomplete list of user characteristics you may be wondering how you’re going to get all this information. There are a number of ways.

The most sound way is through research.

We might learn more about our users by conducting a series of user interviews and consolidating information across interviews.

Information can also be obtained by observing user during usability tests, conducting focus group discussions, or by learning in context in an apprenticeship style interview with your users.

We can also get information by conducting studies or leveraging studies conducted by others. You might get usage details by examining logs from existing software. You might also conduct surveys of your target users directly asking them questions.

All of these approaches, and many I likely haven’t thought of, yield valuable information that can be consolidated, distilled, and placed into relevant profile characteristics.

**Profile characteristics come from assumption**

Making choices based on assumption is often considered a bad idea. But we do it all the time. In fact our assumptions are often based on our past experience and observation. It’s just that we may not recall the specifics of the observation well enough to label it “sound research.” If your company is in the business of writing software for a specific business area or market, there’s likely people in your company that know quite a bit about it. You may be even one of them. Interview these subject matter experts, or talk to yourself if you’re the expert, and collect assumptions about your users to populate your user profiles.

When using assumptions to populate user profiles, make notes about those assumptions and the profile characteristics you’ve used them. For each characteristic, consider how relevant that characteristic is to the design. From here you can make a determination of how risky it is to use an assumption. For an important characteristic, one that might have big impact on your design decisions, you might want to replace assumptions with research.

Leveraging assumptions to create profiles is a quick way to get started. Replacing assumptions with research later helps you reduce the risk of bad design decisions without sacrificing pace too much.
Personas

Personas represent a user constituency with a description of a tangible archetypal user of the system. In addition to specific relevant details, the persona will have a name, picture, and often quotes or stories told in first person by the persona. Create personas based on information gathered to support a user profile. Use personas to more tightly focus design decisions and more concretely communicate to others who your target users are and how your product can satisfy their needs.

Personas become a more tangible design target by selecting characteristic profile data for use in creating a fictional person representative of a class of users.

Actors and roles seem a bit abstract. It’s harder to put a face on them, to understand details about them that may have bearing on the design of a product. Profiling an actor or role begins to add this additional detail. But, an profile can still be somewhat abstract. For instance, a profile may refer to 18-27 year, men or women, with varying computer experience, and varying economic backgrounds. That variance may make it tougher to make design decisions.

Creating a person remedies this by selecting specific details from a profile and using those details to create a archetype of a specific fictitious person that "personifies" the group of users within a user constituency. A profile composed of 18-27 year old men and women with varying computer experience and economic backgrounds might be simply personified by the figure on the left.

Jen, is a 23 year old college student from Philadelphia

Jen comes from an upper middle class background, but like many college students needs to be financially conservative since she’s not yet working. Jen’s been using computers since her family purchased its first Apple computer when she was 7. Today she comfortably moves back and forth from Apple to Windows based computers.

Selecting specific details focuses design

It’s difficult to design for a large group. Trying to satisfy everyone is the road to disaster. Using a persona to target your design allows you to focus on satisfying a single individual. The goal is a strong coherent design that subsequently satisfied most people in its primary user constituency. This rather than a design that looks like it’s trying to make allowances for too many types of people.
Select representative details

For a group of users within a category of information, there will be some variance. But, it’s common to see clusters or areas where a higher density of people within a similar range exist.

Let me be more concrete.

Let’s say we’re designing an application used by largish organizations to process accounts payable. The typical accounts payable clerk’s computer experience may vary wildly from absolute novice to expert. But, if we were to arrange this group on a curve, we might find there are a large number with moderate computers skills, a reasonable number of novices, and very few experts. A curve might look something like this.

Choose representative profile characteristics to use in your persona.

When personifying this group, it makes sense to describe a person with moderate computer skills – someone who’s pretty comfortable using a computer, but not the person you’d call on when you can’t get your laptop to connect to the Starbuck’s WiFi.

Choosing a specific data point from this group of users will focus your design so it doesn’t cater to absolute novices, or proficient experts, and is most likely to work well for the great majority of AP clerks in your target user constituency.

Create personas by making relevant representative information concrete

Create a persona by leveraging profile information and choosing a representative subset of information from each profile characteristic. Connect this representative information together using concise prose. Describe your person in a way that makes them seem concrete.

As you describe your representative person, keep in mind the information needs to be relevant to the design of this product. Including information about the car they drive might be relevant to the design of a handheld product often used in a car, but irrelevant to an accounts payable processing system.

Keep the persona short and easily consumable by its reader.
Make the persona sticky with a name, pictures, quotes, and stories

Give your persona a name. This allows you to refer to him or her in first person. Not only does this help consumers of the persona better identify, but it'll shorten lots of conversations you'll be having about this user constituency. Instead of saying business people who travel often, you can say "Duncan." Choose a memorable name – but not too out of the ordinary.

Add a photograph into the persona to give this persona a face. Let your reader look into the personas eyes.

You can select an image from a variety of sources. When I’m building something for internal use, I’ll fall back on a Google image search. But, if I suspect the image might get out of my company, and I’m concerned with potential licensing issues for the photo, I’ll rely on a source for clip-art images. Currently I use iPhoto.

Select a photo that shows more than your persona’s face. Show your persona in context where they’re likely to spend time using the product you’re designing. A photo of a traveling businessman using a laptop in an airport is a more powerful image than a close-up of a man that looks like he could be a businessman.

Attach quotes to your persona delivered in first person by him or her. Select a quote where the persona says something about their goals, pain points, or needs. You might also select a quote that helps illustrate a detail relevant to design.

If we were building time tracking software for Duncan, he might say:

"The half hour wait for the plan Friday afternoon is when I like to get my time and expenses taken care of so I can enjoy the weekend with my family."

If I were building time tracking software for Duncan, I now know he uses it on a laptop, likely in a busy airport, may or may not have internet access, and wants it to be quick enough to use that he can complete it in a few minutes. I may know someone like Duncan. Even if I don’t, I can look into his eyes, hear his voice, and begin to imagine the sort of software that could satisfy him.

I find that if I’ve gathered information by talking to a few Duncans that I’ll have lots of quotes in my notes that I can use. Stitch a couple quotes together that clearly identify user needs in their voice.

Adding in an anecdotal story further makes the persona more tangible and memorable. Use a story told to you by a potential user of your product. As with quotes, you may blend together a couple stories you’ve heard. Choose a story that helps illustrate the need the user has for this product.

When selecting representative profile details, look for details that compel multiple personas

When looking at profile details you may find a wide variance in a certain interesting area. This may be an opportunity to split a profile into multiple personas.

For example, assume we’re building a new service in our college campus website. We’ve profiled students and we see a wide variance in computer experience and skills. Computer skills seem pretty relevant to downstream design choices. I may choose to create two personas. The first may have strong computer skills. The second may be downright computer phobic. We’re then left with the decision of which of these users we consider more critical to the design of this software. You can see also that the value of the persona we create might extend far past the software to give us guidance on how we might publicize or market this new service.
While looking at profile data, look for relevant variances in characteristics as well as interesting correlations. If you’ve helped conduct interviews or gather other research data, you’ll begin to see patterns, or distinctions that make a difference in your design that compel you to create multiple personas for the same type of actor or role.

FUBU

FUBU is a sportswear company founded in 1992. Although the name FUBU might have originally been intended to stand for "five urban brothers united" it's now commonly thought of as standing for "for us by us." The story goes that the five founders noticed that much of the clothing worn by black people in their community was design by large white-owned businesses. "For us by us" came to refer to the concept that the members of that community could best design their own clothing.

Much of software, and much of the most successful software, starts as FUBU. Someone from a community that has a strong need seeks to satisfy that need by creating a new better tool. That person or group of people likely doesn't build a user model to help them empathize with their target audience, because they are their target audience. The software they're building is for them, by them. They can communicate directly and passionately who the user is, and what their goals and pains are, to their wider team creating the product.

However, if the product is successful, people outside a smaller target community begin to purchase the product. These new customers aren’t as well understood by the original founders of the company. Product designed for the new community is not longer FUBU.

In the mean time, the company founders cease to be members of the community the software was created for, and become members of a community of software designers, developers, and business people. Even end-users pulled from within a company to help lead the development of software to be used by their internal community quickly become co-opted into the software development group they work with. They begin to take on the communities values and ideas, and in the process, begin to shed a few of their own.

As a company grows and diversified its user community, user models become more appropriate and necessary to help communicate and enable designers within the software company to better identify and design for target users.

Leveraging former members of a user community that work for your company now can work well for creating user profiles or personas. But, be sensitive to how long the person you're speaking with has been away from that community.

Ironically, FUBU the company now has a large international group of customers. As I write this, FUBU currently has 70 stores in South Korea. I suspect young Koreans currently buying their product weren’t originally in the minds of the companies five founders in 1992. And, like all smart companies, I suspect FUBU today spends considerable effort studying their current customers, and looking to new customers they might appeal to.

Leverage your user models

Communicate user models in a short documents, posters, or on-line documents

To be valuable the user models you create must be consumed and referenced. Create them in a way that can be displayed in a public area and readily accessed electronically.
It's quick to create an ad hoc poster of a user model by hand writing details on poster paper. If you're building a persona, fix photographs to the paper with tape. Draw cartoons or pictures to help make the model memorable.

Create electronic documents that include photos or figures. Post those document in a place they can be readily accessed. Follow best practices for electronic documents by including information about who created the document, when, and where a most current electronic version can be found.

During collaborative work sessions keep user models handy.

**Less is more, except when it's not enough**

You may gather a large amount of information to base user profiles or personas on. That's good. But, when creating profiles, keep them concise and relevant to the design. Unnecessary details are... well... unnecessary. It'll slow you down to consider information not relevant to your design decision.

If a goal of your user model is communication, than extra, unnecessary detail may actually help better communicate who your users are. But, too many of these risks the creation of a wordy profile that won't be read. Inconsequential details are more likely to "turn off" the profiles reader as they struggle to understand how this profile is relevant to the decisions they've observed or those they need to make themselves.

All that said, distill enough detail into your profile to help you make design decisions. The more critical the profile to your design, the more relevant detail warranted.

**Don't identify too many users in your model**

Just as too much detail on a specific user in your user model makes it difficult to consume the model, too many users have the same effect.

A model containing 3-10 types of users works well.

If you feel you need more users in your model, consider breaking the design process up to focus on large sections or modules of the application where each modules might serve 3-10 types of users. Also consider combining users where there distinctions are likely not to have a significant impact on your design decisions.

If you start modeling users, you're more likely to have too many distinct types. But, it might be possible to have too few. You can tell you have too few when it becomes difficult to make design choices because the diversity within the user constituency results in a large number of good choices. Consider splitting users into different groups to separate out the distinctions that have an impact on your design. If you're not already using a persona, consider personifying diverse users to better focus your design efforts on one type of user.

**Identify Focal Users**

All users are not equally relevant to the creation to the design of your software. Throughout the design and development process you'll need to make trade-offs where you spend more time and effort to include or scale up some product features more than others. Spend that extra time and effort on users that will have the greatest impact on your software. These are our Focal users.

When prioritizing user constituencies in your model keep a model containing business objectives at hand. The users that are most critical to achieving business objectives will be your focal users. If you're building a consumer product, the focal user in your model may be the one that makes buying decisions. This person may be assessing the product's ability to meet business objectives on behalf of their organization. Sometimes their organization is their own household.
You can collaboratively identify focal users in your model using a collaborative prioritization approach such as voting. Make sure those voting do so in the context of understood business objectives and understood users in the model. As the context-setting question: "which of the users in this model are most critical to meeting our business objectives?"

It's a good guideline to allow as much as a third of your user constituencies to be considered focal. In a model containing 10 types of users, 3 or 4 might be focal. In a model containing three users, just one would be focal.

When a piece of functionality will be used by both focal and non-focal users, your goal when choosing and designing functionality is to best support focal users without making life too difficult for those that aren't focal. This is similar to Cooper Interactive's approach to identifying primary and secondary personas - where the software must satisfy the primary persona, while still supporting secondary personas.

**Be model type agnostic**

An effective user model can be easily leveraged to identify feature and design alternatives and function as a basis for making good choices. An effective user model communicates a basis for design choices to everyone involved in the design and development process. You'll find that being conforming to a particular model type isn't necessary to be effective.

**Blend actors and roles as necessary.**

I prefer using roles, but I find in some organizations the common name for a type of user is so common that using a role name to refer to that type of user creates more confusion than clarity. In those cases I mix lists of roles with actors, or common user names.

**For focal users, profile or personify.**

Identifying a user type as focal implies you'll need to spend a bit more time focusing on the functionality to support that user and that user's experience with your product. To make better decisions, you'll find that you may need to add detail to an actor or role. Start by profiling. Or, carry through to personifying to make this user even more tangible. It's OK to blend a little. Using photos and quotes in a profile for instance. Narrowly personify a user by starting from a user role and focusing only on the aspects relevant to the role - that is a type of user engaged in fulfilling a goal with your product.

In the end you may find that you have a couple personas, a profile or two and a few roles annotated with a couple relevant details. That's fabulous so long as you have the information you need to make decisions and adequately communicate your user model to others.

**Identify goals that compel use**

I've seen lots of confusion about exactly what a goal is with respect to software. Much of the time the confusion is my own.

For example if I were writing email software, I might consider a goal of a user using that software to "send email messages."

While that's true, and pretty obvious, it's not the sort of goal we're looking for in a user model. Ask "why would this person send email messages?" and you might arrive at goals like "keep in touch with family," or "communicate with coworkers."

If we know the software is mostly to keep in touch with family, it might compel us to put features in the software to identify how frequently I contact specific family members - and maybe let me know when it's been a long time since I've contacted a family member - to tell me when I'm out of touch.
If we know the software is for communication with coworkers, it might compel us to better thread back-and-forth email conversations, track specific times sensitive messages were read, or to prevent very sensitive communications to be forwarded to others.

For the purpose of modeling users, be concerned with the types of goals that motivate the use of the software. Focusing on these sorts of goals will best help you identify features and feature characteristics that help users reach their goals.

**Buyers, adopters, and users**

At times we confuse people making buying decisions about our software with people using it. You may find this especially true if you’re relying on sales or marketing for intelligence on your users.

When making a software buying or adoption decision a person will likely look at the product relative to other products or choices that person has. For buyers the number of features relative to competitors is important, as are making sure you have the most sought after features. Esthetics are also important. Buyers may make decisions based on very little or no contact with the software. It’s common to make quality judgments based on esthetics.

For some products those in a buying role may never be users. For instance, those making choices to buy enterprise class software will look at features the software has and the expected benefit to their organization. The actual use of the software will occur by others in various roles throughout the company.

In these cases the role of adoption is may also be less significant. The accounts payable clerk may have no choice but to adopt the software purchased by the company’s CEO. However, slow grumpy adopters may impact the return on investment the CEO gets from his or her buying decision.

In consumer products where the buyer, adopter, and user may actually be the same person, it’s important to be aware of changes in concerns as a person moves through buying, adoption, and day to day use.

The features significant to in making a buying decision my not be important in day-to-day use. For example, I may choose to purchase accounting software partly because of its cool graphing functionality, but then never choose to use it. While adopting the software I may expect it to work like other software I’m accustomed to. I might expect quick access to help or tutorials.

When creating a user model, consider the impacts of buyers, adopters, and users. It may be important to separate out a distinct buyer role so that you can clearly identify why you’re making some feature choices in the software. Identifying a focal buying user in your model helps deal with the objection "no one will really use this feature!" They may not be likely to use the feature, but your product may not be purchased or adopted without its inclusion.

**Further reading**

Cockburn, *Writing Effective Use Cases*, Addison-Wesley, 2000
Constantine and Lockwood, *Software For Use*, Addison-Wesley Professional, 1999
Example Profile: Conference Speakers

# of users:
40 or 50 speakers selected from a pool of hundreds

General responsibilities or activities
Often speak at several conferences a year. They review and select conferences to submit to, make submissions, and if accepted submit handouts and slides.

Computer skills
Generally excellent – although some are stronger speakers than computer users

Domain expertise
Strong at their area of expertise. Some are very experienced at applying to and speaking at conferences. Many are infrequent and first time speakers that need a bit of help and advice in submitting and preparing for talks and tutorials.

Goals
Conference speakers want clear instruction on what topics the conference would like to see, how to submit, and how speakers are compensated for participating. They’d like the opportunity to share ideas and in some cases promote their visibility in a particular community to support book sales.

Pain Points
They often miss submission dates and due dates for handouts.

Usage Contexts
Many will use the software from home, and many will use it from the road while traveling.

Software Ecosystem
They’re a mixed batch of mostly PC users but a growing number of Mac users. Generally proficient in Office-type productivity tools and presentation tools.

Collaborators
They often collaborate with each other to prepare talks and socialize. They’ll talk somewhat frequently with conference staff to prepare for and ask questions about an upcoming conference.

Frequency of use
Heavy use a couple times a year, then many months without use at all

Design implications

Feature Opportunities
Email reminders with iCal Calendar entries, downloadable presentation templates, on-line submission status, on-line counts of tutorial registrants

Design Imperatives
Quick to find important dates and venue information. Easy to learn because of infrequent use. Verbose information must be downloadable for later reference.
Example Persona: Jutta

Frequent Conference Speaker
“I really appreciate efficient conference organizers – the ones that value my time.”

Jutta has an over-stuffed conference schedule speaking at over a dozen conferences a year internationally. She travels to US conferences from Germany where she lives. She has one published book and is working on her second. Speaking at conferences allows her to share her ideas with others, promote her work, and network with colleagues to share information and experience.

Over the years Jutta has learned the idiosyncrasies of various conference presenters. Some are more efficient than others. She appreciates those that are early with reminders for submission dates and forthcoming with information she needs to put together submissions. She’s on some conference website every month – but they all vary a bit and it’s frustrating to find the critical information she needs on a particular website she only sees once every few months.

She’s been using computers since she was young, and although much of her work is done talking with people and facilitating collaborative work, she’s proficient on the Windows based applications she spends time on every day.

Feature Opportunities
Email reminders with iCal Calendar entries, downloadable presentation templates, on-line submission status, on-line counts of tutorial registrants

Design Imperatives
Quick to find important dates and venue information. Easy to learn because of infrequent use. Verbose information must be downloadable for later reference.
Blank Persona 1
Blank Persona 2