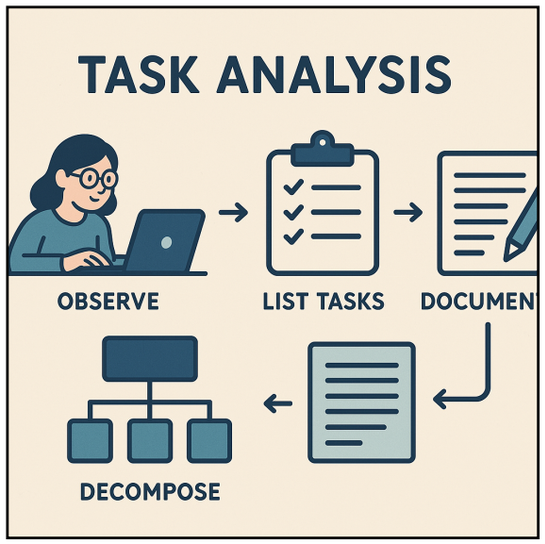
Task Analysis

Harley Eades



Today we're diving into Task Analysis — a fundamental tool in human-centered design. This lecture will focus on how people actually perform tasks in the real world and how we, as designers, can support those tasks effectively through our interfaces. Keeping with my theme of badly designed AI photos, here is the one for task analysis. This one isn’t as bad as the others unfortunately.

Task Analysis

Focus on how do people accomplish a specific tasks

Helps identify the tasks that your solution must support

Helps to find effective ways of accomplishing a task

Task analysis helps us understand how people accomplish things. It gives us insight into the steps they take, the tools they use, and the challenges they face. This information is crucial because it tells us what our system needs to support—and just as important—how to support it well.

Task Analysis

Use in combination with other user research methods

Task Analysis is a lens on the information you obtain through other user research methods

In practice you should iteratively decide how to best draw upon all relevant methods throughout a process

This isn’t a standalone activity. Task analysis should be used alongside other user research methods. Think of it as a lens—we analyze user research findings through this lens to uncover the sequence and structure of actions users take. And in practice, this is an iterative process—you return to it as your design and your understanding evolve.

Task Analysis: Questions to ask

Who is going to use the system?

What tasks do they now perform?

What tasks are desired?

How are the tasks learned?

Where are the tasks performed?

What is the relationship between people & data?

What other tools do people have?

How do people communicate with each other?

How often are the tasks performed?

What are the time constraints on the tasks?

What happens when things go wrong?

To conduct a task analysis, we ask a lot of questions—who's using the system, what do they want to do, how do they learn it, where do they do it, what happens when things go wrong, and so on. These questions form the foundation of understanding user behavior in context.

Who is going to use the system?

Identity:
  In-house or specific customer is more defined
  Broad products need several typical consumers

Background

Skills

Work habits and preferences

Physical characteristics and abilities

Let’s take that first question: Who is going to use the system? The answer may be very specific—say, dental assistants in a small clinic—or broad, like general consumers. We want to learn about their background, skills, preferences, even physical characteristics. All of this influences design.

What tasks do they now perform?
What tasks are desired?

Important for both automation and new functionality

Relative importance of tasks?

Observe people, see it from their perspective

Automated Billing Example
- small dentists office had billing automated
- assistants were unhappy with new system
- old forms contained hand-written margin notes
- e.g., patient's insurance takes longer than most

Next, we compare what tasks users currently perform with what they want to do. Sometimes our goal is automation, sometimes it's enabling new functionality. Observing users is key here.

Take the example of billing automation in a dental office:
the new system didn't support margin notes, which were crucial.
This highlights why we need to see tasks from the users' perspective.

In addition, users don't always realize how they complete a task, and so we need to watch carefully to uncover the little nuances they might not notice; like the margin notes.

How are the tasks learned?

What does a person need to know to perform the task?

Do they need training?
- academic
- general knowledge / skills
- special instruction / training

How are tasks learned? Do users need training or special knowledge? Are tasks intuitive or do they require prior instruction? The answers here impact not just the interface design but the overall user experience—especially onboarding.

Where are the tasks performed?

- Office, laboratory, point of sale?
- Effects of environment on customers?
- Are people under stress?
- Confidentiality required?
- Do they have wet, dirty, or slippery hands?
- Soft drinks?
- Lighting?
- Noise?

Context matters. Is the task done in an office, a noisy restaurant, a lab with slippery surfaces? Environmental factors like lighting, noise, or stress level can radically change how people interact with technology. Always consider the physical setting.

What is the relationship between people & data?

Personal data
- Always accessed at same machine?
- Do people move between machines?

Common data
- Used concurrently?
- Passed sequentially between customers?
- Remote access required?
- Access to data restricted?
- Does this relationship change over time?

Next, we think about how users interact with data. Is the data personal or shared?

For personal data, we need to know how it is access:
- always on one?
- or do people move between machines?

For shared data (common data):
- Is remote access needed?
- Are there restrictions?
- How does these permissions change over time?

These questions affect how we design data flows and access permissions.

What other tools do people have?

- More than just compatibility
- How customer works with collection of tools

Automating lab data collection (example):

- how is data collected now?
- by what instruments and manual procedures?
- how is the information analyzed?
- are the results transcribed for records or publication?
- what media/forms are used and how are they handled?

Users rarely operate in a vacuum. They work with a collection of tools. Understanding how these tools interoperate—or don't—is vital. If you're designing software for a lab, you need to understand how existing instruments and procedures fit into the workflow.

How do people communicate with each other?

- Who communicates with whom?
- About what?
- Follow lines of the organization? Against it?

We also need to know how people communicate. Who talks to whom? About what? Is communication formal or informal? Systems should support the actual communication patterns, not just the org chart.

How often are the tasks performed?

- Frequent use likely remember more details
- Infrequent use may need more help
  - Even for simple operations,
  - Make these tasks possible to accomplish

- Which function is performed
  - Most frequently?
  - By which people?
- Optimizing for these will improve perception of performance

How often is a task performed? This affects memory load and usability. Frequent tasks should be optimized for speed and efficiency. Infrequent tasks may need extra guidance or support—tooltips, wizards, or even just simpler flows.

What are the time constraints on the tasks?

- What functions will people be in a hurry for?
- Which can wait?
- Is there a timing relationship between tasks?

Are users in a rush? Are tasks time-sensitive? Sometimes a task can wait—sometimes it's urgent. Think of a nurse entering patient data versus a casual social media post. Timing affects prioritization.

What happens when things go wrong?

- What happens when things go wrong?
- How do people deal with
  - task-related errors?
  - practical difficulties?
  - catastrophes?
- Is there a backup strategy?
- What are the consequences?

This is one of the most important questions: what happens when things break? Are there error recovery paths? Is there a backup plan? Systems should help users recover from errors gracefully—because mistakes will happen.

Selecting Tasks

- Real tasks people have faced or requested
  - collect any necessary materials
- Should provide reasonable coverage
  - compare check list of functions to tasks
- Mixture of simple and complex tasks
  - easy tasks (common or introductory)
  - moderate tasks
  - difficult tasks (infrequent or for power use)

When deciding which tasks to analyze, focus on real ones—things people actually do. Include a mix of simple, moderate, and complex tasks. And make sure these tasks cover all the important functionality of your system.

What should tasks look like?

- Say what **person** wants to do, but not how
  - allows comparing different design alternatives
- Be specific, **stories** based in concrete facts
  - say who person is
  - design can really differ depending on who
  - give names (allows referring back with more info later)
  - characteristics of person (e.g., job, expertise)
  - story forces us to fill in description with relevant details
- Sometimes describe a complete "**accomplishment**"
  - forces us to consider how features work together

Task descriptions should say what the person wants to do—not how they do it. Be specific and concrete. Give the person a name, a background. Turn the task into a story—this helps us really understand what the user needs and what success looks like.

Using tasks in design

- Write up a description of tasks
  - formally or informally
  - run by people and rest of the design team
  - get more information where needed

> Manny is in the city at a restaurant and would like to call his friend Sherry to see when she will be arriving. She called from a friend's house while he was in the bus tunnel, so he missed her call. He would like to check his missed calls and find the number to call her back.

We use these tasks to guide interface design. Sketch rough interfaces, make sure each feature supports a real task. If it doesn't—ask whether that feature is really necessary. Or maybe you need to add a task to justify it.

Let's look at a concrete example. Manny wants to call Sherry back but missed her call. What tasks does this involve? Checking call history, retrieving her number, making the call—all while in a potentially noisy public place. How do we support that smoothly?

Using tasks in design

- Hierarchical Task Analysis
  - focused on decomposing a high-level task into subtasks
- Cognitive Task Analysis
  - focused on understanding tasks that require:
    - decision-making
    - problem-solving
    - memory
    - attention
    - judgement

https://www.interaction-design.org/literature/article/task-analysis-a-ux-designer-s-best-friend

Tasks can be analyzed in two ways: hierarchically or cognitively. The former is concerned with breaking tasks down into subtasks so that each piece can be considered in our design. The latter is focuses on tasks that require decision making, problem solving, and/or the use of memory, attention, or judgment.

Task: Park in a new neighborhood

Peter is going to brunch on a Sunday with his roommates. He is trying a new place he found on Yelp. He has the address for the place and he is using a smartphone GPS for directions. He leaves the apartment with his roommates at around 8:30am and he wants to beat the crowd so they won't have to wait in line. He is driving a Toyota Corolla that he has owned for five years. It is a rainy day and he doesn't have an umbrella.

Let's consider an example of hierarchical task analysis. Read the example. How do we break this task down?

Task: Park in a new neighborhood

unknown neighborhood/restaurant

Peter is going to brunch on a Sunday with his roommates. He is trying a **new place he found on Yelp**. He has the address for the place and he is using a smartphone GPS for directions. He leaves the apartment with his roommates at around 8:30am and he wants to beat the crowd so they won't have to wait in line. He is driving a Toyota Corolla that he has owned for five years. It is a rainy day and he doesn't have an umbrella.

First, we can see that Peter is going to an unknown neighborhood/restaurant. Also…

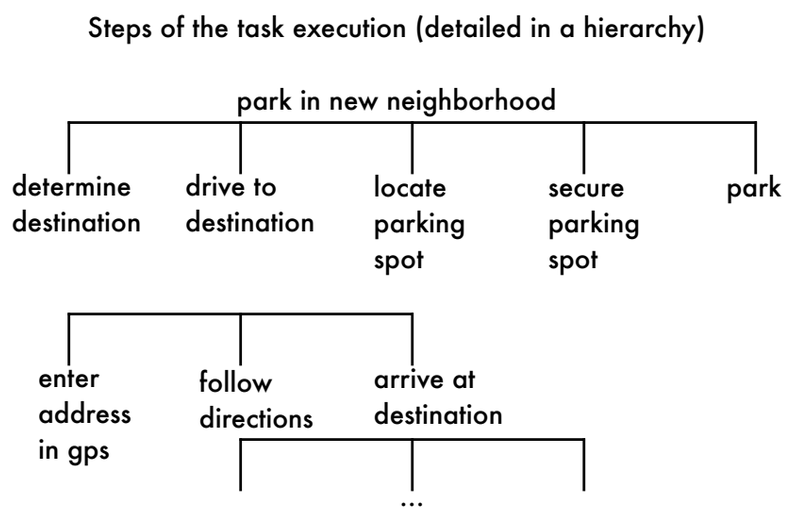
Task: Park in a new neighborhood

needs to find a parking spot close by?

Peter is going to brunch on a Sunday with his roommates. He is trying a new place he found on Yelp. He has the address for the place and he is using a smartphone GPS for directions. He leaves the apartment with his roommates at around 8:30am and he wants to beat the crowd so they won't have to wait in line. He is driving a Toyota Corolla that he has owned for five years. **It is a rainy day and he doesn't have an umbrella.**

It is raining outside, and he doesn't have an umbrella. Using…

Hierarchical Task Analysis: Park in a new neighborhood

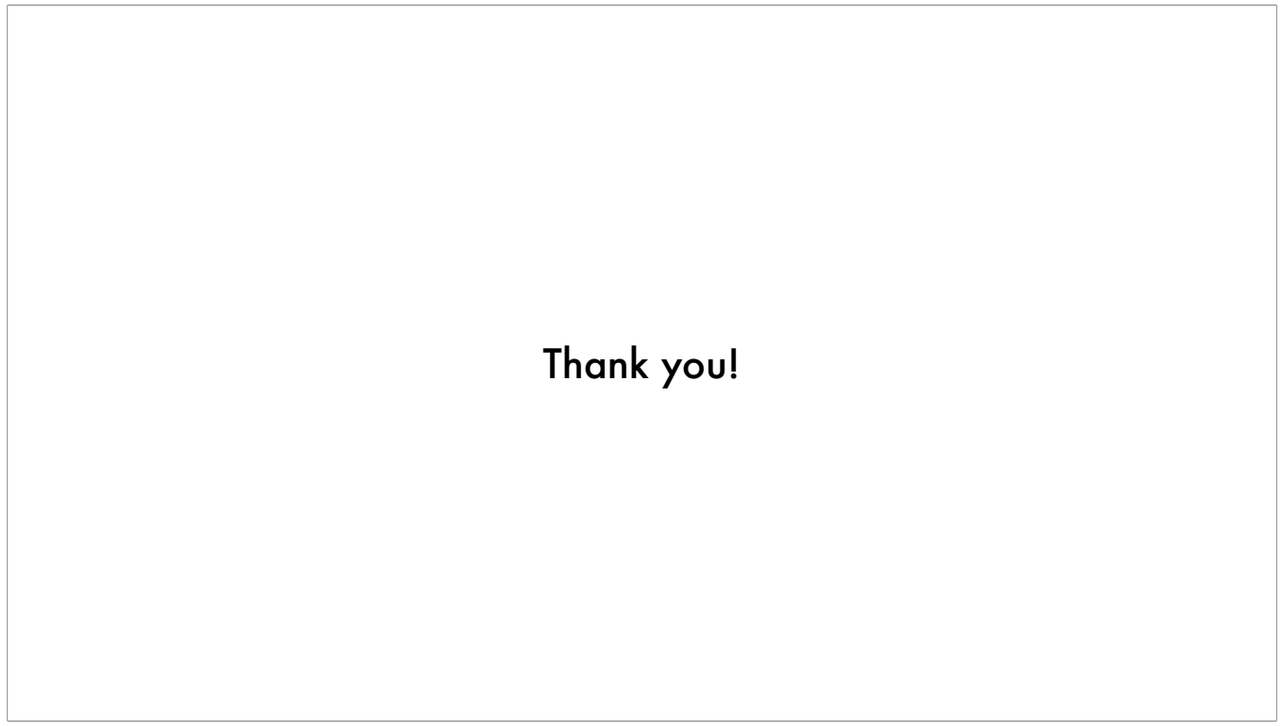


Hierarchical task analysis we can break this down into subtasks as follows. Read through the slide.

Using Tasks in Design

- Rough out an interface design
  - discard features that do not support your tasks
  - or add a real task that exercises that feature
  - major elements and functions, not too detailed
  - hand sketched
- Produce scenarios for each task
  - what person does and what they see
  - step-by-step performance of task
  - illustrate using storyboards

We use tasks to rough out interfaces. Sketch a scenario—what the user does, what they see. Use storyboards to visualize each step. This helps align the design with actual user behavior and ensures all features are grounded in real needs.



And that wraps up our session on Task Analysis. Remember: your goal is to support what users are really trying to do, in their context, with their tools, under their constraints.