MENTAL MODELS

ALIGNING DESIGN STRATEGY WITH HUMAN BEHAVIOR

BY INDI YOUNG

Rosenfeld Media
Brooklyn, New York

Appendix B:

The Evolution of the Mental Model Technique

The first time I desperately needed to explain the function of a product from the user's point of view to a group of product managers was in 1993. The project was headed into the abyss of feature overload, and I, the young programmer with a whole six years under her belt, felt that I could turn the ship around. I was designing a new call center application for Visa as a part of a large team that was bringing a new-fangled graphical user interface, "GUI," along with updated hardware, to the reps in the center. "Gooey!" the product managers crowed. I cringed. I couldn't even say the word then,[1] even when it was popular. I had a duty to set them on the right path.

[1] Remember, 1993 was a time when many corporations still sent internal communications by memo, typed up in Microsoft Word, printed, and delivered to physical mailboxes in the copy room on every floor. People were still excited about the new "desktop metaphor" on their personal computers.

Visa Call Center

Before we arrived on the scene, the reps in the Visa call center were using a menu-based, text-only interface. They loved the system because once they memorized the exact key strokes it took to hop over unneeded fields and get from one page to another, they were fairly fast. They would occasionally get up from their chairs, unplug their headsets and wander through the cube maze with the cords dangling over their shoulders, walking to the machine that printed out directions to ATMs in any country. This trek was thought of as a kind of a break in the routine. The frantic tone of conversation with desperate card holders experiencing all the emotions of someone who has been robbed in a foreign city was exhausting. A literal Babel of voices filled the room, with reps fielding calls in 13 languages. The unending pace of calls coming in from the Voice Recognition Unit (VRU) could be unnerving. Schedules were strict. Security was tight. There were a lot of students working in the call center. It was considered a temporary job.

The team was made up of software engineers like me, Visa employees with business backgrounds who had developed the requirements, and a group of consultants in charge of setting up the hardware system. Together we were working through a requirements document that was 321 pages

long. I was responsible for generating and testing window-based user interface screens for the reps. Not only was I grappling with issues of training and data entry speed, but we were all overwhelmed by the depth and detail of the requirements. As was customary, the requirements and diagrams were designed from the system's point of view (as shown in Figure B.1).

In an effort to represent all the requirements from the documents in a workflow diagram, I drew pictures in which the rep's tasks were highlighted (Figure B.2). But these ended up looking a lot like the system-oriented diagrams.

Since I had some experience with relational database design, one of my tasks was to guide the team setting up the database tables. The goal of a relational database design is to eliminate duplication of data across tables, which is called "normalization." I was reviewing the tables one afternoon in a cubicle when I realized I could normalize the rep's tasks, too, into a task model. In this manner, I would be able to distill all those documented requirements into a neat set of instructions.

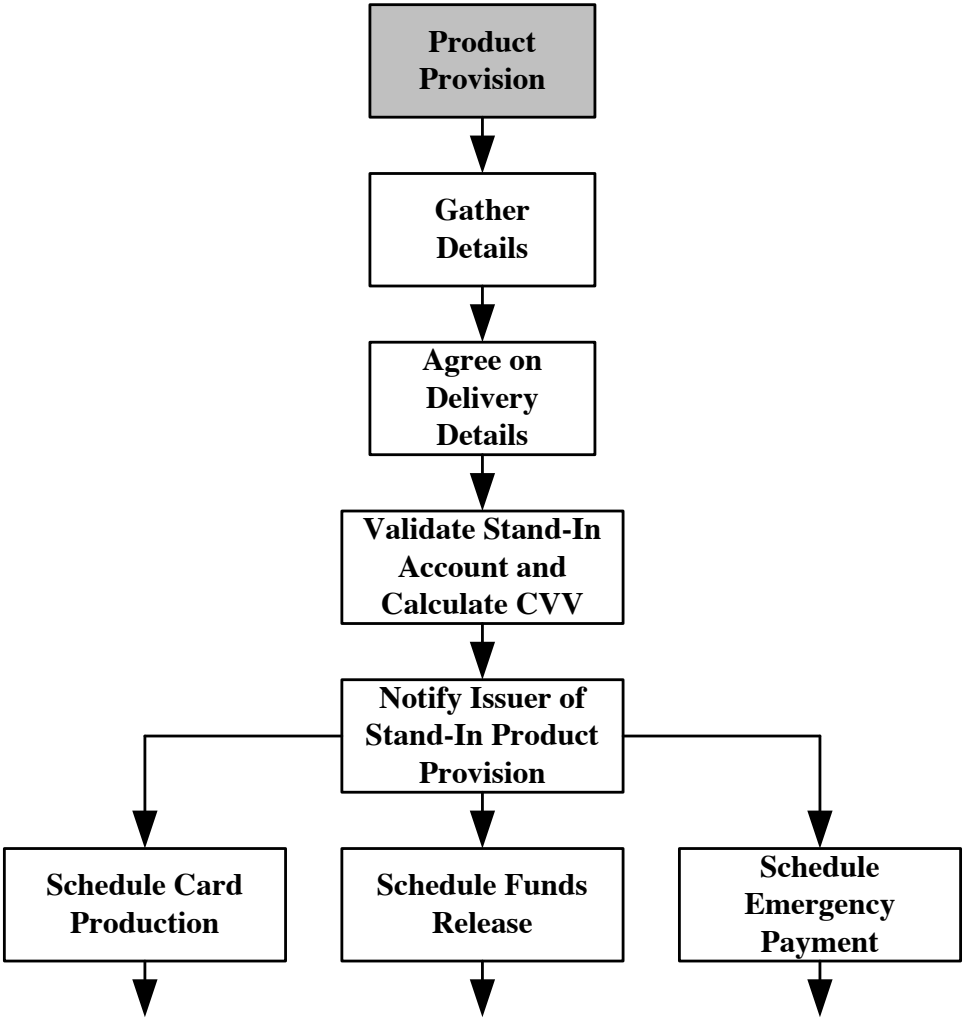

FIGURE B.1.
Typical workflow diagram from the system perspective.

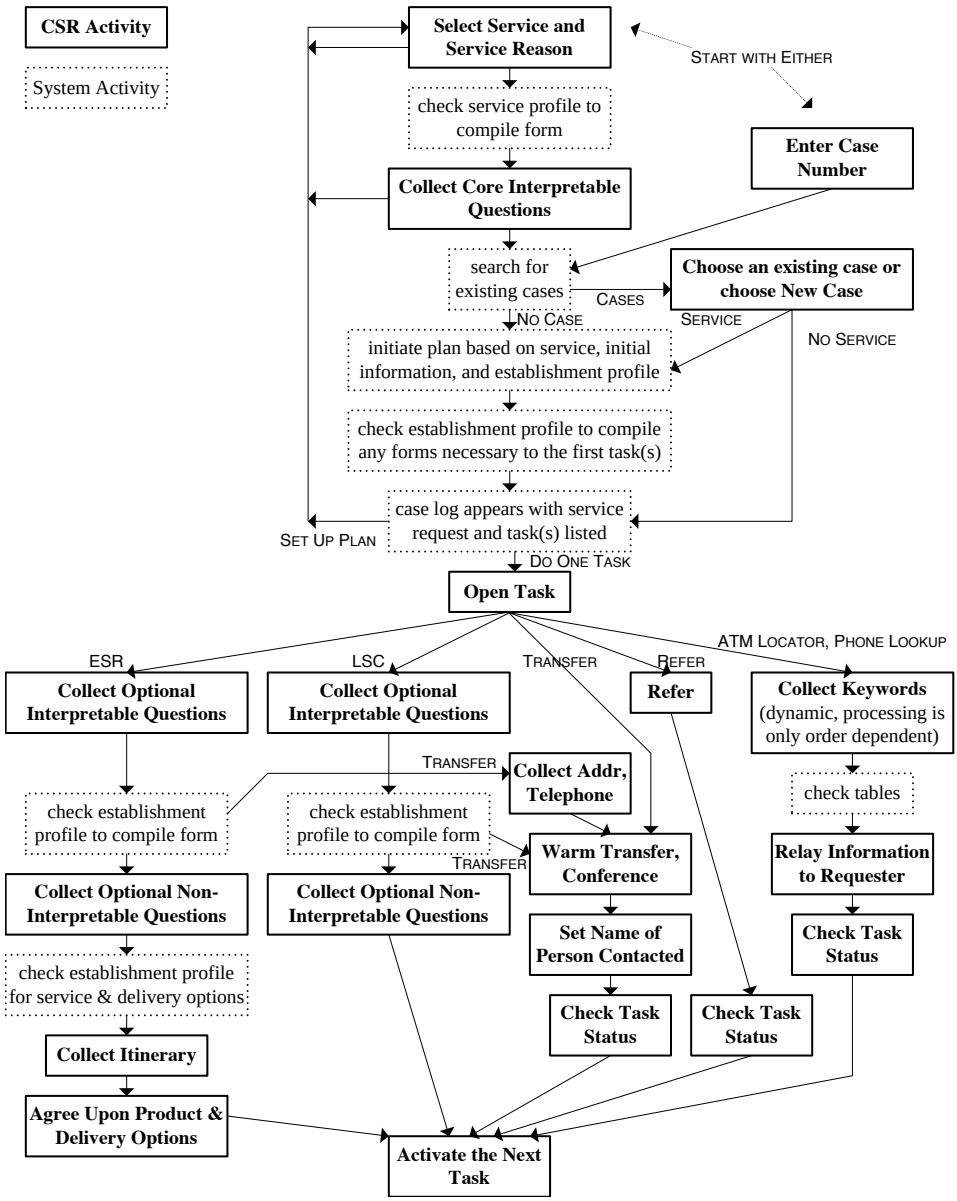

FIGURE B.2.
Workflow diagram with rep tasks in solid-line boxes.

I quickly sketched out a table format that I thought could hold rep tasks. Then it dawned on me that I had the building blocks of a finite state machine, one of those constructs that they teach you as a computer science major. It models the transition from one state to another based on actions within the system. People who write parsers for compilers use these models. I decided I would model the actions and transitions of state for a call center rep.

I came up with the task tables in Figure B.3. You use these by first identifying a task that needs to be done, such as on row two, "Verify Account Status," a system task. Using the Task ID "XF2," you look up conditions in the Next Task Table to find the next step. Say the conditions are that the verification is complete and the issuer prefers fax contact. You would see the next task for XF2 is NI1. The Task Table lists NI1 as "Send Account Detail" to the issuer, another system task. If the condition for this task is a busy signal, and more than 20 minutes have passed since

the inception of the task, the next task is NI2, which is voice contact by a CSR—a task completed by the system user.

TASK TABLE
(Bold means no corresponding entries in the Next Task table yet; italic means not a CSR task)

TASK ID	TASK	DATA SET/MANIFESTATION	RESP.	MED.	TIME LIMIT
XF1	Send	Account Status	System	Link	30
XF2	Verify	Account Status	System	Link	30
NI1	Send	Account Detail	System	Fax	30
NI2	Send	Account Detail	CSR	Voice	30
NC1	**Query**	**Account Detail**	**CSR**	**Voice**	**480**
NC2	**Send**	**Account Status**	**CSR**	**Voice**	**30**
NC3	**Send**	**Account Status**	**CSR**	**Fax**	**30**
CI1	**Verify**	**Network Status**	**CSR**	**Voice**	**30**
CI2	**Verify**	**Exception File**	**CSR**	**Voice**	**30**
CI3	**Verify**	**Fax Machine**	**CSR**	**Voice**	**30**
CI4	**Verify**	**Contact Numbers**	**CSR**	**Voice**	**30**
CI5	**Verify**	**Fax Machine**	**CSR**	**Voice**	**30**
PC1	Send	Embossing Request	System	Email	60
PC2	Verify	Embossing Request	CSR	Voice	90
PC3	**Send**	**Embossing Request**	**CSR**	**Voice**	**60**
PC4	*Produce*	*Card*	*PRC*		*2880*
NC2	**Query**	**Itinerary**	**CSR**	**Voice**	**480**
SD1	Send	Delivery Request	System	Fax	30
SD2	Verify	Delivery Request	CSR	Voice	60
SD3	**Send**	**Delivery Request**	**CSR**	**Voice**	**60**

FIGURE B.3.

NEXT TASK TABLE

TASK ID	LINK STATUS	OPTION	NEXT TASK	AT TIME
XF1	Link Is Down	If <= 20 minutes have passed	XF1	5
XF1	Link Is Down	If > 20 minutes have passed	CI1	0
XF1	Complete		XF2	5
XF2	Link Is Down	If <= 25 minutes have passed	XF2	5
XF2	Link Is Down	If > 25 minutes have passed	CI1	0
XF2	Account Status Not Equal	If <= 10 minutes have passed	XF1	0
XF2	Account Status Not Equal	If > 10 minutes have passed	CI2	0
XF2	Complete	Issuer prefers fax contact	NI1	0
XF2	Complete	Issuer prefers voice contact	NI2	0
NI1	Busy Signal	If <= 20 minutes have passed	NI1	5
NI1	Busy Signal	If > 20 minutes have passed	NI2	0
NI1	No Answer	If <= 20 minutes have passed	NI1	5
NI1	No Answer	If > 20 minutes have passed	NI2	0
NI1	No Fax Answer		NI2	0
NI1	Incomplete Transmission	If <= 15 minutes have passed	NI1	0
NI1	Incomplete Transmission	If > 15 minutes have passed	CI3	0

FIGURE B.3. (CONTINUED)
Task tables in which you look up what needs to be done, then identify conditions and read off the next task from the second table.

Suddenly, I had a Task Table that encompassed the tasks of the rep and the call center system we were building, as well as tasks for producing and delivering the emergency Visa card. I showed the table to the software engineers, and they loved it. The consultants in charge of the hardware system said I had done all their work for them. They could see the whole system. The product

managers and business folks, however, needed some coaching to get through the table. Several had MBAs and were quick to follow the function of the table and the value of being able to see it all on one page, but since it was mechanical, it was cumbersome for them to read. Unfortunately, these were the people I most wanted to win over to user-centered software design.

By 1996, I was referring to the task tables as "task analysis" because I had heard the term used by some speakers at a conference. For a company called Colloquy[2], I wrote what I called a UI Model for a collaborative knowledge management system. By this time, I had realized the need to move out of the system mindset entirely to design within the real world of the user, reaching beyond the computer-based tools they might use.

---

2 Colloquy was renamed Intraspect later in 1996, and was then acquired by Vignette in December 2003. See http://tomgruber.org/technology/intraspect.htm

This perspective was explained in the following paragraph in the UI Model that I wrote:

"What is it that people do when they collaborate? Since there are a few computer-based collaboration applications in use now, we can analyze the tasks available to users in the application menus and toolbars. Additionally, there are plenty of corporations still collaborating the pre-electronic way, often as an adjunct to software applications. We can catalogue activities from the more common world. ...We want to list a complete set of types of knowledge management. We want this set to be as broad as possible so that our application does not overlook any small, but important, uses of collaboration."

I was exploring tasks for two reasons: one, I wanted to have a complete understanding of what functions the application needed to support, and how they should be organized; and two, I wanted to ensure that the business strategy—the reason we were building an application—mapped to the breadth of real-life collaborative knowledge management.

So I started cataloging tasks and activities. I wrote the tasks as verb+noun pairs, and described each task with a use-case paragraph. Then I listed a set of sub-tasks that made up the larger task. This was my way of organizing all the hundreds of tasks I found in literature or existing applications. Here is one example:

Write a Research Paper

Svoboda is writing a paper to help her biotech company obtain venture capital, and looks for all related papers and results to use as references or related readings. She reads all the papers in detail to understand what was reported. She asks her peers how they felt about the findings in each paper. She records where each paper was published and looks to see if the study was discredited in subsequent issues of the journal.

- Find publications related to a topic
- Say what you think of a publication
- Find out what other people think of a publication
- Find out whether a publication is creditable

Then I took each sub-task and listed as many atomic tasks as possible. I did this by thinking of all the tasks associated with each

medium or tool the person might use. Here I explore one sub-task, "Say what you think of a publication."

Say what you think of a publication:

- Publish your opinion of the document in your own web publication (public and private servers)
- Post your opinion of the document to a discussion (threaded discussion)
- Email your opinion of the document to someone you know (email)
- Tell people your opinion of the document (face-to-face meeting)
- Present your opinion of the document to an audience (live/recorded presentation)
- Tell people your opinion of the document (telephone/voicemail)
- Handwrite your opinion on the document itself (manually distributed paper)

This was a top-down approach, guided by the researcher: me. It was an effort to delineate the structure of the tasks from a high level, and then fill in as many sub- and atomic-level tasks as possible. It bothered me that

I was still doing this from my ivory tower (office cube) based on what I read, heard, deduced, and discussed with co-workers. I really felt the need to flip this approach on its head and start with atomic tasks, preferably collected from the users themselves—rather than from a limited set of brains at the office. In subsequent projects, I gathered tasks by reaching out to the users.

I came to the same conclusion that Alan Cooper later discussed in his 1999 book: *The Inmates are Running the Asylum*. Those of us creating applications don't think much about how it will work, just how we will build it.

A Real "User Perspective"

Because "task analysis" was an off-putting term for most of the product managers and business folks I worked with, I began to refer to the task lists as a "mental model" of the user. It was a representation of what users were trying to accomplish and how they could accomplish it, looking beyond just the tool design.

The mental model was only one of two things that I wanted, however. The second was an understanding of the breadth of real-life situations, so I could make sure it supported the business strategy. In 2000 I managed to supply this piece by making the tri-level task list a diagram, and mapping application features to the various components of the diagram. I was working on a project for a company that sold geospatial image data collected by satellite to various engineering and military clients.[3] By then I was actually collecting task data by interviewing people in the real world. I had interviewed 16 people representing two groups: those who bought and used geospatial data, and those who supplied parts of the whole picture. I did my task analysis and drew a diagram that compared the tasks of the users/buyers to the tasks of the suppliers/VARs (Figure B.4). I used the resulting diagram to recommend verbally to the company what groups of

3 This was long before Google Earth, when everyone got access to this data and some of the more curious and inventive people started mashing it up with other types of data to tell stories.

tasks (vertical sections in the diagram) not to pursue in the course of doing business.

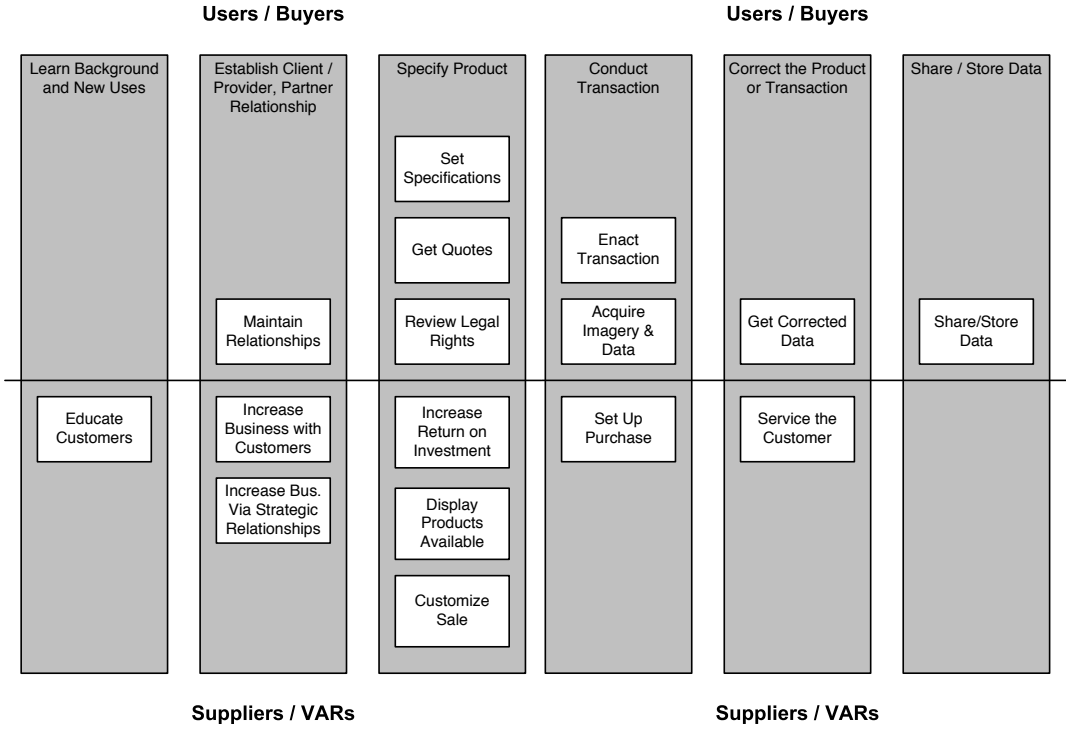

FIGURE B.4.
Task alignment among audience segments.

The next time I did this type of research I went ahead and added the application features as boxes directly in the diagram, matching them to tasks in the user mental model. I laid out the mental model horizontally above a thick line. Then I put the application features (tools,

guides, various data views, etc.) into boxes the same size as the task boxes so they would line up nicely in vertical columns. Then I moved the feature boxes below the horizontal line under each set of tasks that they supported. The project was for a large financial firm which allowed its customers to manipulate their accounts online. It was January 2001 when I created this diagram:

User Mental Model

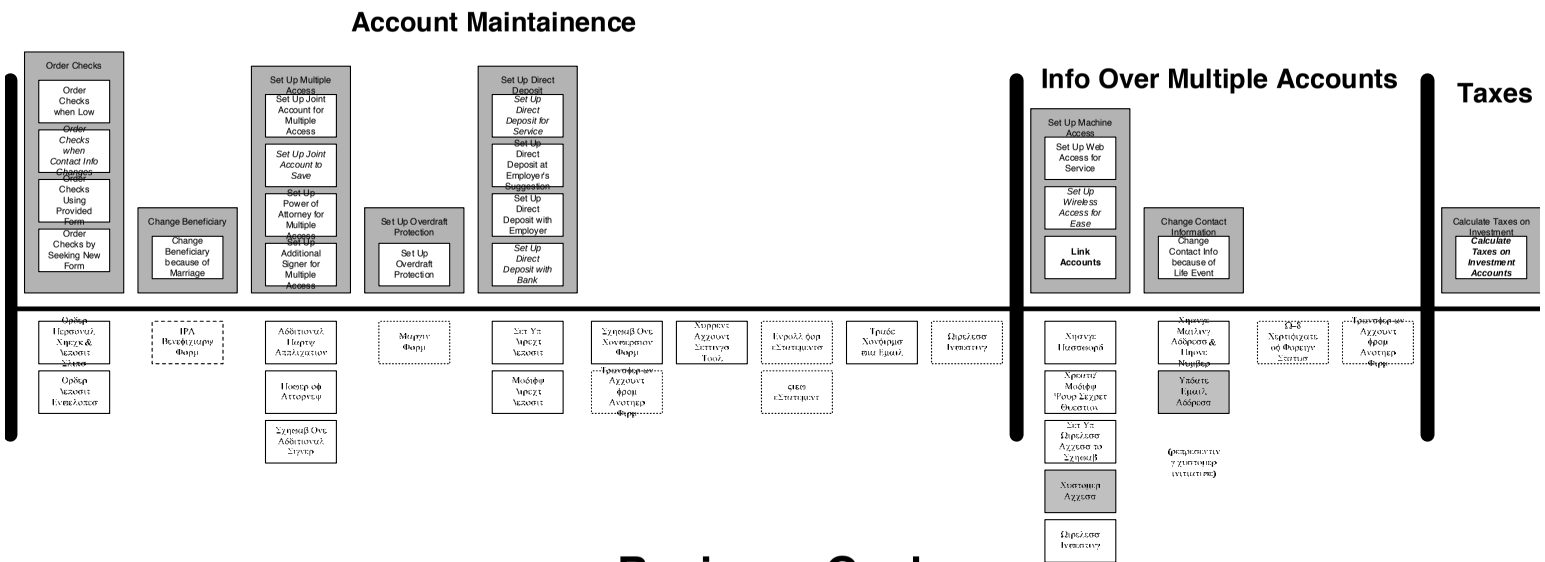

FIGURE B.5.
First mental model between tasks and features. (Image is deliberately small to protect the client's data.)

The diagram met with great success inside the financial firm. People all the way up to the executive level could easily understand the diagram. By matching the business requirements to the mental model, we could see where there were opportunities to empower the user with different features. I had finally achieved my second goal.