Conceptual Models in a Nutshell

by Jeff Johnson and Austin Henderson

January 22nd, 2013 5 Comments

Jeff Johnson

Jeff Johnson is Principal Consultant at UI Wizards, Inc., a product usability consultancy (uiwizards.com). He also is a principal at WiserUsability.com, a consultancy focused on elder usability and accessibility. After earning B.A. and Ph.D. degrees from Yale and Stanford Universities, he worked as a UI designer and implementer, engineer manager, usability tester, and researcher at Cromemco, Xerox, US West, Hewlett-Packard Labs, and Sun Microsystems. He has taught at Stanford University, Mills College, and the University of Canterbury. He has authored many articles and chapters on Human-Computer Interaction, as well as the books GUI Bloopers, Web Bloopers, GUI Bloopers 2.0, and Designing with the Mind in Mind. His latest book, coauthored with Austin Henderson, is Conceptual Models: Core to Good Design (2011).

Austin Henderson

Austin Henderson's 45-year career in Human-Computer Interaction includes user interface research and architecture at MIT's Lincoln Laboratory, Bolt Beranek and Newman, Xerox Research (both PARC and EuroPARC), Apple Computer, and Pitney Bowes, as well as strategic industrial design with Fitch and his own Rivendel Consulting & Design and Scalable Conversations. Austin has built both commercial and research applications in many areas. These applications, and their development with users, have grounded his analytical work, which has included the nature of computation-based socio-

This article explains what conceptual models are and describes the value of developing a conceptual model of a software application before designing its user interface.

Conceptual Model: a Model for Users' Mental Model

A conceptual model of an application is the model of the application that the designers want users to understand. By using the application, talking with other users, and reading the documentation, users build a model in their minds of how to use the application. Hopefully, the model that users build in their minds is close to the one the designers intended. This hope has a better chance of being realized if the designers have explicitly designed a clear conceptual model as a key part of their development process.

A conceptual model describes abstractly — in terms of tasks, not keystrokes, mouse-actions, or screen graphics — what users can do with the system and what concepts they need to be aware of. The idea is that by carefully crafting a conceptual model, then designing a user interface from that, the resulting application will be cleaner, simpler, and easier to understand. The goal is to keep the conceptual model: 1) as simple as possible, with as few concepts as are needed to provide the required functionality, and 2) as focused on the task-domain as possible, with few or no concepts for users to master that are not found in the application's target task domain.

Object/Operation Analysis

An important component of a conceptual model is an Object/Operation analysis: an enumeration of the user-visible object-types in the application, the attributes of each object-type, and the operations that users can perform on each object-type. Purely presentational and purely implementation object-types have no place in an application's conceptual model because users will not have to be aware of them.

Objects in the conceptual model of an application can usually be organized in a type-hierarchy, with sub-types inheriting operations from their parent types.

technical systems, the interaction of people with the technology in those systems, and the practices and tools of their development and use, particularly the conversations that surround them. The primary goals of his work have been to better meet user needs, both by improving system development to better anticipate those needs, and by improving system capability to enable users themselves to better respond to unanticipated needs when they arise in a rich and changing world.

Stories by Jeff Johnson

Conceptual Models in a Nutshell
5 Comments

Most-commented Stories

The Power of Collaboration
1 Comments

HTML's Time is Over. Let's Move On.
105 Comments

The Lazy IA's Guide to Making Sitemaps
63 Comments

Blasting the Myth of the Fold
58 Comments

The Age of Findability
57 Comments

Depending on the application, objects may also be organized into a containment hierarchy, i.e., in which some objects contain other objects. Laying out these two hierarchies in a conceptual model greatly facilitate the design and development of a coherent, clear user interface.

This analysis can help guide implementation, because it indicates the most natural hierarchy of implementation objects and the methods each must have. It also simplifies the application's command structure by allowing designers to see what operations are common to multiple objects and therefore can be designed as generic operations. This, in turn, makes the command structure easier for users to master: they must only learn a few generic commands that apply to many object-types, rather than a larger number of more narrowly applicable object-specific commands.

For example, in a well-thought-out application that allows users to create and manipulate both Thingamajigs and Doohickeys, when users know how to create a Thingamajig and want to create a Dohickey, they already know how because creation works the same way for both. Ditto copying, moving, deleting, editing, printing, etc.

Example: Object/Operation Analysis for a Simple Office Calendar App

For example, let's examine an objects/operations analysis for a simple office calendar application. The objects, attributes, operations, and relationships might be as follows:

Objects: It would include objects like **calendar**, **event**, **to-do item**, and **person** (see Table 1). It would exclude non-task-related objects like *buffer*, *dialog box*, *database,* and *text-string*.

Attributes: A **calendar** would have an *owner* and a *default focus* (day, week, month). An **event** would have a *name*, *description*, *date*, *time*, *duration*, and a *location*. A **to-do item** would have a *name*, *description*, *deadline*, and *priority*. A **person** would have a *name*, a *job-description*, an *office location*, and *phone number*. However, **Events** should not have *byte-size* as an exposed attribute, because that is implementation-focused, not task-focused.

Operations: **Calendars** would have operations like *examine*, *print*, *create*, *change view*, *add event*, *delete event*. **Events** would have operations like *examine, print,* and *edit*. **To-do items** would have more-or-less the same operations as **events**. Implementation-related operations like *loading* databases, *editing* table rows, *flushing* buffers, and *switching* modes would not be part of the conceptual model.

Objects	Attributes	Operations

Calendar	owner, current focus	examine, print, create, add event, delete event
Event	name, description, date, time, duration, location, repeat, type (e.g., meeting)	examine, print, edit (attributes)
To-Do item	name, description, deadline, priority, status	view, print, edit (attributes)
Person	name, job-description, office, phone	send email, view details

Table 1. Object/operation analysis for a simple office calendar application.

Keep it Simple

Sometimes it is tempting to add concepts to provide more functionality. But, it is important to realize that each additional concept comes at a high cost, for two reasons: 1) it adds a concept that users who know the task domain will not recognize and therefore must learn, and 2) it increases the complexity of the application exponentially, because each added concept interacts with many of the other concepts in the application. Therefore, extra concepts should be resisted if possible. The operative design mantra with conceptual models is: “Less is more.”

A Conceptual Model Provides a Foundation for the App and the Project

The user interface design translates the abstract concepts of the conceptual model into concrete presentations and user-actions. For best results, the user interface is designed *after* the conceptual model has been designed. Scenarios can then be rewritten at the level of the user interface design. Designing the UI from the conceptual model may expose problems in the conceptual model, in which case the conceptual model may be improved.

A conceptual model provides a foundation not only for the UI design, but also for the application’s implementation and documentation. It therefore plays a central role in the design and development of the overall product.

Summary: Six Benefits of Conceptual Models

Starting a design by devising a conceptual model has several benefits:

1. By laying out the objects and operations of the task-domain, it allows designers to notice operations that are shared by many objects. Common operations across objects make the UI simpler for users to learn and remember.
2. Even ignoring the simplification that can result from noticing shared operations, devising a user-model forces designers to consider the relative importance of concepts, the relevance of concepts to the task domain (as opposed to the computer domain), the type hierarchy of objects, and the containment hierarchy of objects. Having thought about these things greatly facilitates designing a user-interface.
3. A conceptual model provides a starting point for the development of a product vocabulary, i.e., a dictionary of terms that will be used to identify each of the objects and operations embodied in the software. This helps ensure that terms are used consistently thoughout the app and its documentation.
4. Once designers have a conceptual model for an app, they can write scenarios depicting people using the app to perform tasks, using only concepts from the conceptual model and terms from the vocabulary. For example, a conceptual-level scenario for the calendar application might be: "John checks his appointments for the week. He schedules a team meeting, inviting team members, and adds a dental appointment." Such scenarios (which can be separated into *use-cases*), help validate the design in functional reviews. They can also be included in product documentation and training. Conceptual scenarios describe tasks and goals without revealing the UI-level user interactions required to achieve those goals, so they can be used as task descriptions in usability tests.
5. A conceptual model provides a first cut at the app's object-model (at least for the objects that users will be aware of), so developers can use it to begin implementing the app.
6. An actively-maintained conceptual model supports a better development process. It can insure that all user-visible aspects of an application (functionality, terminology, UI, documentation, support, ...) are *consistent*. By making the conceptual model the joint responsibility of all team members, the application can be made *coherent*. Both of these also *reduce development resources* by reducing rework.

Further Reading

• Johnson, J. & Henderson, D.A., "Conceptual Models: Begin by Designing What to Design", *Interactions*, Jan-Feb 2002.

• Johnson, J. & Henderson, D.A., *Conceptual Models: Core to Good Design*, Morgan & Claypool, 2011.

Tags: conceptual models

Tags: conceptual models
Posted in Design Principles | 5 Comments »

5 Comments

Chris Stone January 23, 2013 at 3:58 pm

Great summary Jeff! I've often approached my projects through this lens but never as clearly expressed as this post. It reminds me that I can apply this approach to every project, big or small, product or service.

Thank you sir.

Hans Verhaegen January 24, 2013 at 8:57 am

Great read, thanks! Small remark: I think you switched the attributes and the operations for 'person'?

Patrick January 25, 2013 at 4:03 pm

Great read. I think I kind of do this naturally but not in such a thought out, organized manner. Great stuff to consider.

Jeff Johnson January 27, 2013 at 9:09 pm

Thanks Chris, Hans, and Patrick for your feedback.

Hans: Yes, in Table 1, the attributes and the operations for 'person' were switched. The editors of this blog have fixed it now. It turns out that the error was also in our book, Conceptual Models: Core of Good Design. I've let the publisher know, and they will correct it there too. Fortunately, the book is sold mainly as an eBook, allowing errors to be corrected more quickly than if the book were in print.

Faye Miller January 28, 2013 at 3:49 am

Nice article, Jeff. As a researcher it is helpful for starting to think about how to communicate design with developers to collaborate on creating a user

interface.

Leave a comment

Name (required)

Mail (will not be published) (required)

Post Reply