

Architecture tip: User interface design using the OODA loop

A checklist for getting your user interface design off to the right start

Skill Level: Introductory

[Tom Hammell \(tshammell@us.ibm.com\)](mailto:tshammell@us.ibm.com)
Staff Software Engineer
IBM

12 Mar 2008

The OODA loop was developed by the Air Force as part of its study of air combat. This observe-orient-decide-act (OODA) cycle models the human brain's decision-making process. This article explains how UI architects can use the ideas of the OODA loop to design user interfaces (UIs) that give users better situational awareness and are more intuitive to use. In this article, learn the basics of the OODA loop, and find out how it can be applied to UI design. The end of this article features a checklist you can use to help with the initial design of your UI.

What is the OODA loop?

The Air Force has studied air combat since World War I. Over the years, different theories of air combat have evolved and culminated with the development of the OODA loop by John Boyd.

Figure 1. The OODA Loop

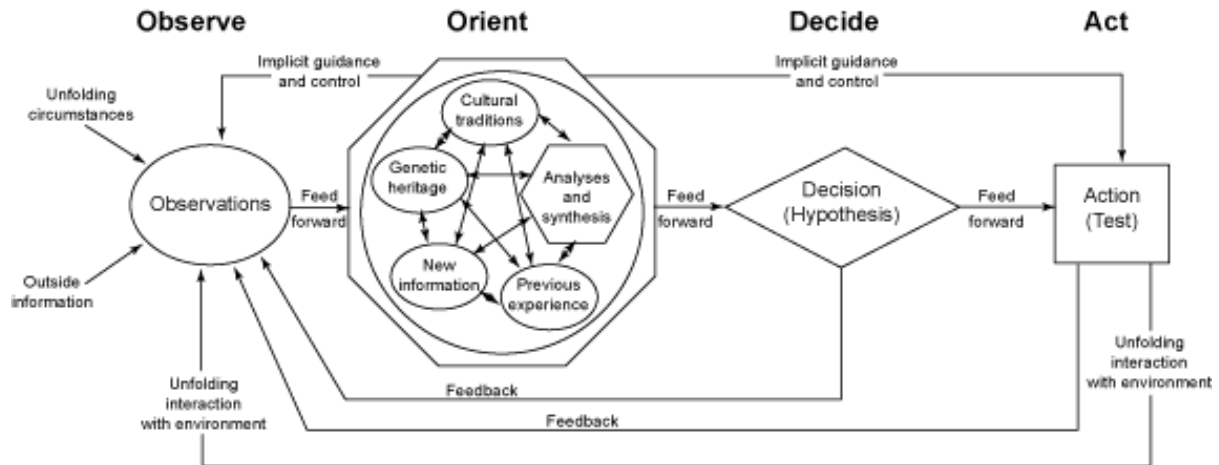


Figure 1 shows the basic steps in the loop and how they are related. The OODA process is complex and a full explanation is outside the scope of this brief article, but the pictures offer a good overview. The key to understanding the OODA loop is understanding how information is fed back and forward between each step and how all the pieces work together as a whole. The basic steps can be defined as follows:

- **Observe**—Gather information from the environment about the problem that you are trying to solve.
- **Orient**—Use the information gathered to form a mental model of the circumstances. That is, synthesize the data to assess the situation in your mind. As more information is received, you deconstruct old mental models, then create new ones to better fit the situation. Note that different people require different levels of details to perceive an event. We often imply that the reason some people cannot make good decisions is that those people are bad decisions makers. However, the real reason most people make bad decisions is that they fail to place the information that they do have into its proper context. This is where orientation comes in. The orientation step should take into account the users' experience and skill level. It should also emphasize the context in which events occur, so the user can make better decisions on how to act. Orientation is what helps turn information into knowledge. And knowledge, not information, is the real predictor of good decision making.
- **Decide**—Consider options and select a course of action that you think will help you solve the problem.
- **Act**—Carry out the conceived decision and test the result. Once you observe the result of the action, you feed back this information to the beginning of the loop and start the process over again. Note that in combat (or in competition), you want to cycle through the four steps faster and better than the enemy, which is probably working through its own OODA loop.

How to use the OODA loop for UI design

So what do air combat and the OODA loop have to do with UI design? A lot, actually. The OODA loop is a good model for how people interact with their environments and how they make decisions to solve problems. A UI is simply a piece of software that helps a user solve a problem—whether that problem is how to find a book from Amazon.com or how to control a nuclear reactor.

A good UI should help people quickly go through the OODA process. First, it should show them the information that they need to solve a problem (allow them to *observe*). Next, it should present the information in a way that makes it easy for them to understand the problem (*orient* them to the situation). Then it needs to show them the possible actions that can be taken (present them with the opportunity to *decide*). Finally, it should provide feedback on the results of the action that was taken (enable them to see the results of their *actions*).

A well-designed UI should allow a user to carry out tasks as quickly and efficiently as possible. There have been many studies about UI design and how to improve it. The result of these studies is usually a set of guidelines to use when you create a UI. The guidelines often dictate how to format different types of data or what color and font to use for text. Although these types of conventions give a consistent look and feel to the UI and improve usability, they do not provide much help in the initial design of the UI.

If you look at a UI as a piece of software used to make a decision, you can use the OODA loop as the model for the decision process when you're designing a UI. Just put yourself in the users' position and try to create a set of screens that will help them gather information and make decisions as fast as possible. Look at the UI as an implementation of the OODA loop, and come up with a list of questions that you should ask at each step in the design process to help the user quickly make decisions.

To start the UI design process, the first question that you should ask is

What is the problem that this UI solves?

This question is important because the answer helps you, as a designer, determine the domain of the problem. The description of the problem to be solved should be as specific as possible. Something like, “The user needs to monitor the temperature of a nuclear reactor to ensure it stays below a certain range, and the user needs to be able to shut down the reactor if the temperature is above the temperature range,” is a very specific answer to the problem.

Observe

After you identify the problem that the user is trying to solve, the next question to ask

is

What data does the user need to understand the problem domain?

The answer to this question will determine what information needs to be gathered from the problem domain. It may require collecting information from a database, querying a Web service, or getting data from a previous user action. It is important to make sure you have a good idea of all the information that a user would need to fully understand the problem. The information to collect includes external data from the environment as well as internal information fed back from the other parts of the UI.

Orient

Now that you have gathered the information, the next question to ask is

How can I present the data in a way that will give the user a comprehensive picture of the problem domain?

You need to determine how to organize and present the data. This will involve reformatting and combining data to present it in a model that will give the user a clear mental picture of the problem domain. When determining how to present the data, you need to take into account the user's previous knowledge and experience. That brings up another set of questions:

What type of user will be using the UI? What type of experience and knowledge does the user have about the problem? How does the user turn information into knowledge?

You should find out the level of familiarity the user has with the system, what type of experience the user has, and so on.

This picture should be constantly updated as the underlying data changes, so the user understands not only the current state of the problem domain but also how changes affect the domain.

The key here is to understand how the user views and interacts with the domain so that you can present it in a format that is familiar to the user.

Decide

The model presented to the user may be very simple or very complex. If the model is simple—for example, a table with a list of invoices that need to be paid—the decision can be as simple as, "Pay the invoices that are due." If the model is complex—perhaps a diagram showing a nuclear reactor going critical—the decisions on how to prevent a meltdown will be complex. The UI should provide guidance on these decisions.

The question here is

What actions are associated with each part of the model?

The answer to this question will help you determine how to add actions (buttons, context-sensitive menus, check boxes, and so on) that will let the user know what actions are associated with each part of the model. The various actions enable users to make decisions that will affect the model.

Act

After the user acts, it is important that he or she receives feedback on the result of the action. If the user paid an invoice, you may want to display a message saying the invoice has been paid and then update the table, marking the status of the invoice as "paid." The question to answer here is

After an action has been taken, how should the model be updated to show the result of the action?

The answer to this question is critical to success of the user interface. The quicker the user understands the results the actions taken, the sooner the user can understand whether he or she has solved the problem or needs to take more actions.

After you have answered questions for each part of the OODA loop, the last question to ask is.

What information should be fed forward or fed back between the different parts of the UI to help improve the user's knowledge of the state of the model?

The UI should present a model to the user that is consistent with his or her particular view of the problem being solved. As outside information changes and as the user takes actions, the changes need to be reflected in the model as quickly as possible. Quickly presenting changes to the model will improve the user's situational awareness and allow the user to make better decisions.

If you can answer all of these questions in sufficient detail, you can use the answers to create use cases, documentation, and a number of other artifacts needed in the design and implementation of the UI.

Of course there is much more to the process of UI design. Answering the questions will get you started on the design, but you will need to continue to refine it. Refining is where usability practices like defining a consistent look and feel, creating prototype screens, and getting user feedback come in. If you take the time to think about the answers to the questions and follow some simple usability practices, you should be well on your way to creating a good UI.

UI designer's checklist

Developers working on UI design should try to understand the OODA loop and use it in the initial design of the UI. The focus should be on understanding the problem that the UI is solving and providing UI that helps the user make quick decisions. If you use the questions in the checklist to help guide your design, you should be able to design a UI that users will enjoy using.

Checklist questions

1. *What is the problem that this UI solves?*
2. *What data does the user need to understand the problem domain?*
3. *How can I present the data in a way that will give the user a comprehensive picture of the problem domain?*
4. *What type of user will be using the UI? What type of experience and knowledge does the user have about the problem? How does the user turn information into knowledge?*
5. *What actions are associated with each part of the model?*
6. *After an action has been taken, how should the model be updated to show the result of the action?*
7. *What information should be fed forward or fed back between the different parts of the UI to help improve the user's knowledge of the state of the model?*

Resources

- Check out the [Wikipedia entry](#) for more detailed information on the OODA loop.
- Visit the [User Interface Engineering site](#) for general UI design tips.
- "[Strike a balance: Users' expertise on interface design](#)" by Mike Padilla (developerWorks, September 2003) examines what makes an application UI usable and details concepts that can facilitate an efficient, broad-based UI design.
- Discover the power of customer guidance when designing a Web application user interface in "[Learn from your customers for usable Web apps](#)" by Paul Englefield (developerWorks, June 2003).
- "[Debunking the myths of UI design](#)" by Paul Smith blasts the myths around user interface design and its costs and benefits (developerWorks, February 2002).
- "[Reducing the user interface](#)" by Mark Molander (developerWorks, June 2001) shows you how to filter your way to reduced UI data and UI functions.

About the author

Tom Hammell

Tom Hammell is a staff software engineer with IBM and works on developing software for enterprise data management. He has more than 20 years of experience developing software. He has published numerous articles on Java™ topics, ranging from Swing development to unit testing and has published a book on test-driven development. He also lectures frequently on Java topics. He holds a bachelor's degree in electrical engineering and a master's degree in computer science from Stevens Institute of Technology.