



Application Note

Bare Essentials: A Note on Simplifying User Interfaces by Simplifying Use Cases

Larry L. Constantine
Constantine & Lockwood, Ltd.
University of Technology, Sydney

Abstract: *This brief application note illustrates how reducing the number of steps in the essential use cases within a task model can simplify the user interface and contribute to more efficient task performance by users.*

Keywords: essential use cases, essential models, usability, user interface design, simplification, user efficiency, task modeling

Learn more about usage-centered design at <http://www.forUse.com>.

We are often asked about the importance of reducing use cases to their most simplified, abstract, and generalized form. How important is it? Simplifying use cases to their most essential form is one of the keys to success in usage-centered design. It can give developers a leg up on good user interface design. By eliminating unnecessary or "technology-driven" steps, designers are guided to user interfaces that simplify and speed the most common or basic operations.

A particularly clear example arose in a training class at one of our European clients. The problem involved the design of a touch-screen control panel for the "Thumper 2000," an industrial materials-testing machine that pounds things to pieces. (I'll leave out a lot of the details for the sake of discussion here.) One of the focal user roles is the **Standard-Test-Running Role**, basically a machine operator who knows little or nothing about materials, engineering, or testing, but who can set up the machine mechanically and then initiate and monitor any of a series of predefined test sequences. Under supervision or limited special circumstances, an operator in this role might make small adjustments to a standard test protocol.

Supporting the **Standard-Test-Running Role**, among others, is a focal use case we can call **running predefined test**. One possibility for the essential use case narrative looks something like the one shown below. (Step number 5 refers to a user-initiated optional extension, a use case defined separately.)

<u>running predefined test</u>	
USER INTENTIONS	SYSTEM RESPONSIBILITIES
1. request standardized test	2. display list of standardized tests
3. pick desired test	4. display chosen test setup
5. optionally [do <u>adjusting test setup</u>]	6. prepare machine for test as setup and confirm readiness
7. confirm start	8. run test
	9. report results

Unlike orthodox use cases as used in object-oriented software engineering, essential use cases can begin either with a user intention or a system responsibility. One design team in the training class pointed out that the use case above can be further simplified by eliminating the first user step, leading to the use case shown below.

<u>running predefined test</u>	
USER INTENTIONS	SYSTEM RESPONSIBILITIES
	1. display list of standardized tests
2. pick desired test	3. display chosen test setup
4. optionally [do <u>adjusting test setup</u>]	5. prepare machine for test as setup and confirm readiness
6. confirm start	7. run test
	8. report results

Why, it was argued, should an operator in the **Standard-Test-Running Role**, who may be ignorant of and uninterested in other operations, always have to start by telling the system that one of the standard tests is intended? Teams working from the longer version of the use case came up with prototype main screens along the lines shown in Figure 1. This is a pretty familiar form of user interface: start with a whole slew of buttons to let (or make) the user choose which task before beginning. We sometimes refer to this as a naïve, one-button-per-use-case design. Like many uninspired conventional designs, it imposes extra steps on every use.

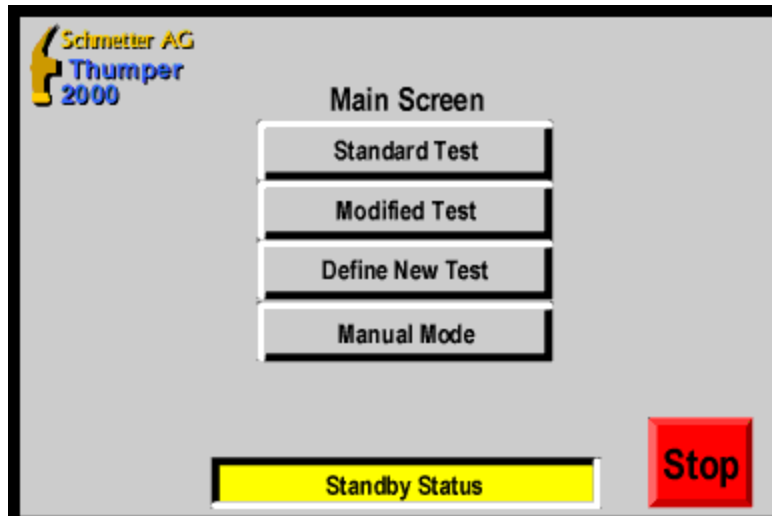


Figure 1 – A naïve design based on unsimplified use cases.

The simpler essential use case implies a different architecture for the user interface. The team that simplified the use case created a prototype design that surfaced the standardized test list at the top level on the main screen. That design was organized along the lines shown in Figure 2 below. The result allows the most straightforward common tasks to be carried out without switching screens.

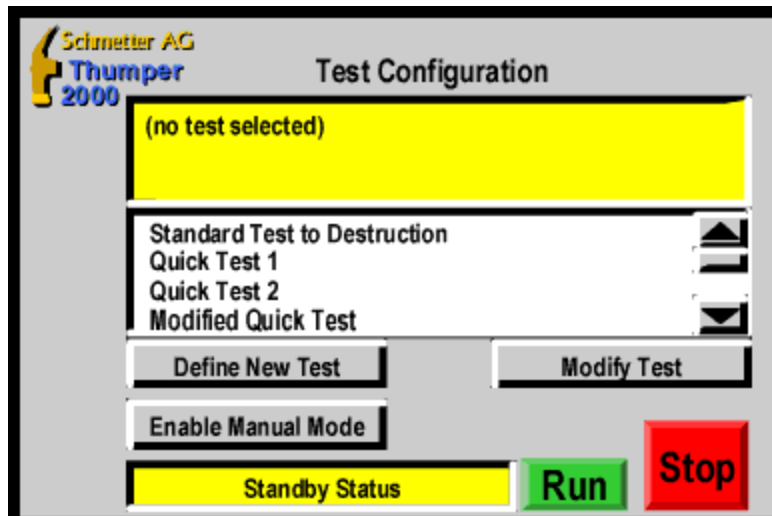


Figure 2 – A more efficient, easier-to-use design based on simplified use cases.

One should not, of course, naively apply this notion to everything. Carried to its absurd extreme, surfacing user interface controls to save user steps would put everything on the initial screen or opening dialogue. All design is a matter of tradeoffs, and the trick is to choose where to draw the line. In this simplified example, the line is drawn at the simplest use cases for the simplest uses. More complex and sophisticated uses, such as the use case **defining new standard test**, are quite reasonably "buried" down an extra level.

In relatively small design problems, an experienced designer might well spot the opportunities for simplified operation with or without good essential use cases.

However, when the stakes are escalated to systems with scores or even hundreds of use cases, the merits of simplifying the use cases down to their barest essentials can be substantial.

Larry Constantine, January 1999

Learn more about usage-centered design at <http://www.forUse.com>.