

---

# Using Optimization to Design Fast, Simple, and Versatile User Interfaces

## Richard C. Davis

School of Information Systems  
Singapore Management University  
80 Stamford Road  
Singapore 178902  
rcdavis@smu.edu.sg

## Abstract

When designing a user interface, teams must often balance conflicting demands for speed, simplicity, and versatility, but their ability to do so is limited. Interface optimization is an analysis technique that helps designers discover trade-offs early in a design process. Designers compare functional specifications by performing usage scenarios on imagined UIs and recording the steps. This data is processed with optimization techniques to produce visualizations of the design space. This paper briefly describes interface optimization, how it fits into a UI design process, and preliminary experiences using the technique.

## Author Keywords

Interface optimization; conceptual models

Copyright 2015 Richard C. Davis

## Introduction

Many applications aim to simplify complex tasks, but they become increasingly complex as new functions are added. In recent years, however, an increasing number of cloud-based and mobile apps (such as Google Docs) are stripping out all but the most essential functions. This raises a question: which functions are essential? The answer is often unclear until a vendor releases a product and observes its acceptance in the market.

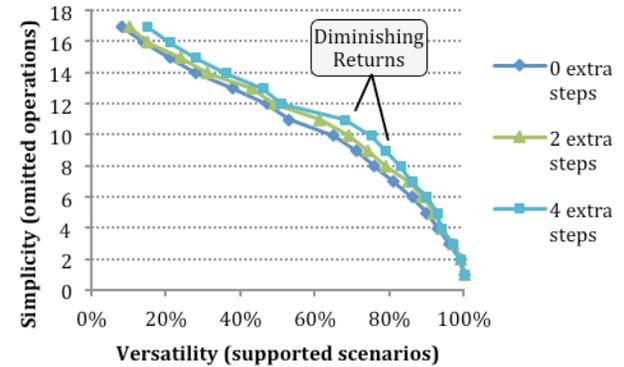
Interface optimization gives software vendors a principled way to choose functions *before implementation begins*. Designers analyze usage scenarios to create a set of candidate conceptual models, which define abstract objects and methods in a user interface (UI) without specifying presentation details [3]. Designers may think about presentation when creating this model, but the lack of specification makes it easy to consider many possibilities. Designers then compare possible conceptual models by performing scenarios on imagined UIs and recording the steps. Scenarios may also be ranked by market data that indicates the relative priority of supporting each scenario. This data is fed into an optimization program to produce visualizations (see Figure 1) that help designers see the speed, simplicity, and versatility tradeoffs in each candidate conceptual model.

The term “optimization” is used for this technique, because a constrained optimization solving engine lies at its core. The uncertain nature of user and market data makes it impractical to compute a single, optimal UI, but optimization results can still help designers weigh design tradeoffs. Optimization can find conceptual models that maximize simplicity given speed and versatility constraints. The end goal is to help designers create user interfaces that are *optimal* in the sense that they are as fast and simple as possible while supporting as many tasks as possible.

Interface optimization has been used in two design projects: an informal animation tool (K-Sketch [2]) and a casual game design tool [1]. This preliminary experience shows that the visualizations produced by interface optimization help designers make decisions by revealing the trade-offs that become obscure as the number of design alternatives grows. Designers can also understand these tradeoffs better, because points in the design space can be linked back to the data that produced them.

### Speed, Simplicity, and Versatility

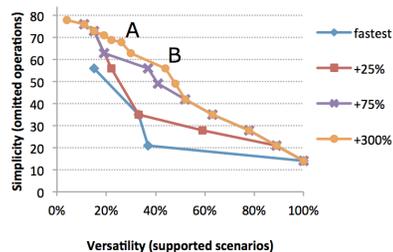
Software vendors consider many factors when choosing the functions for a software product. I separate factors into three categories that influence all UIs: speed, simplicity, and versatility. *Speed* concerns how quickly tasks can be accomplished. The choice of tasks is important. Two word processors may be equally fast for writing a single letter, but if one has a mail merge function, it will be much faster for writing 100 form letters. In interface optimization, designers conceptually perform tasks with a UI’s functions and record the steps. A UI’s speed is reflected in the number of steps needed to perform tasks with those functions.



**Figure 1:** Interface optimization results for K-Sketch. When assuming a speed of 4 extra steps, a model with simplicity score of 10 (i.e., that omits 10 out of 18 possible operations) has a versatility score of 75% (i.e., it supports 75% of scenarios). After this point, simplicity begins to fall rapidly with little added versatility.

*Simplicity* concerns the cognitive effort involved in learning how to perform tasks with a UI. This effort can be modeled as the number of perception-action pairs that a user must commit to memory to perform tasks [4]. Since the number of pairs to memorize increases with the number of functions required to perform tasks, we can approximate this effort simply by counting the number of functions required. One UI can thus be called *simpler* than another if it provides fewer functions. I define a UI’s *simplicity* as  $M - N$ , where  $N$  is the number of functions provided by the UI, and  $M$  is the total number of functions across all candidate UIs.

*Versatility* concerns the breadth of tasks for which a UI is appropriate. Vendors consider market data and



**Figure 2:** Interface optimization data for the casual game design tool showing optimal simplicity and versatility points when we assume scenarios are performed as fast as possible, or 25%, 75%, or 300% longer. Points A and B were chosen for further development.

available resources when choosing the tasks their software will support. Many vendors hope to sell their product to as many people as possible and may attempt to support an aggressively large number of tasks. I model versatility as the percentage of desirable tasks supported by a UI.

While speed, simplicity, and versatility as I have defined them are certainly not the only factors that should influence the choice of functions in a user interface, I claim that they contain useful information for any UI design team. Interface optimization helps designers see how speed, simplicity, and versatility are interrelated. The technique also allows designers to order functions by their importance, helping them to identify functions that should be more visible in the UI or appear earlier in training materials.

### Interface Optimization Phases

I now briefly describe the 5 phases of interface optimization and explain how it has been used:

1. Collect **scenarios** for all potential uses of a UI.
2. Design one or more conceptual models and collect all possible functions into a set of **operations**.
3. **Code** each scenario by recording all **approaches** to performing it as a list of operations.
4. Use optimization tools to **process** coded scenarios.
5. **Visualize** the results.

#### Phase 1: Collect Scenarios

Scenarios can be gathered through interviews, observation, questionnaires, chance meetings, or artifacts created with systems like the one being designed. A set of scenarios should accurately represent what target users will want to do. There should be a scenario for every type of user or task that the system *might* support. For K-Sketch, 72 animation

scenarios were gathered from semi-structured interviews [2]. For the game design tool, 14 games and 13 simulations were gathered.

#### Phase 2: Define Operations

Scenarios can be used to design a conceptual model for a user interface. Interface optimization is most useful when designers have several conceptual models to choose from or when a conceptual model has too many functions. Conceptual models need not be formal; they can be as simple as a list of function names or as complex as a collection of UML diagrams. This technique only requires that functions can be collected into a set of conceptually distinct *operations*. The operation set for K-Sketch can be found in Table 1.

#### Phase 3: Code Scenarios

To *code* a scenario, a designer lists all the different ways to perform it using the operations defined in phase 2. Designers perform scenarios quickly on imagined user interfaces and record the sequence of steps. The results of this process are imprecise but they are sufficient for interface optimization to produce valuable visualizations.

The first step in coding a scenario is to divide it into *elements*, which are parts of a scenario that can be performed independently. In K-Sketch, most elements corresponded to classes of moving objects. In the game design tool, elements also correspond to behaviors. For each element, the designer then identifies the operations that might be used to perform that element. With this, she can enumerate all the *approaches* to performing each element with the given set of operations. An approach is a sequence of steps that performs an element of a scenario. There should be an

Scenarios Supported	68%	75%	79%
Num. of Operations	7	8	9
Num. of Solutions	1	1	3
Translate	1	1	3
Scale	1	1	3
Rotate	1	1	3
Set Timing	1	1	3
Move Relative	1	1	3
Appear	1	1	3
Disappear	1	1	3
Trace			2
Repeat Motion		1	2
Copy Motion			1
Define Cels			1
Morph			
Physically Simulate			
Interpolate			
Move Forward/Back			
Deform			
Move Limb			
Orient To Path			

**Table 1:** K-Sketch optimization results for the 3 data points highlighted as *diminishing returns* in Figure 1 (4 extra steps). Optimization found one set of 7 operations that supports 68% of scenarios, one set of 8 operations that supports 75% of scenarios, and 3 sets of 9 operations that support 79% of scenarios. Each row show the number of sets that an operation appears in. The highlighted cells show the solution that was used to design K-Sketch.

approach for every group of operations that might handle the element in a different way. Approaches are annotated with their speed. In K-Sketch, the annotation was the number of extra steps required (i.e., how many more than the fastest approach). In the game design tool, the annotation was the estimated time taken by the approach.

#### Phase 4: Process Coded Scenarios

Coded scenarios are now processed with optimization techniques to map out interesting points in the design space. This is done by repeatedly solving the problem, “If the UI could have only  $N$  operations, which operations should be chosen to maximize the number of tasks (versatility) that can be completed within  $M$  of the fastest approach’s speed?” This problem is solved for all possible values of  $N$  (simplicity) and for select values of  $M$  that cover the range of speed possibilities.

#### Phase 5: Visualize Results

Two types of visualization have proven useful. The simplest is to plot a curve in the simplicity-versatility plane for each selected speed value. Such plots can reveal points where a relatively small number of operations support a relatively large number of scenarios. In Figure 1, the right-most point in the *diminishing returns* area was chosen for K-Sketch. In Figure 2, the points marked *A* and *B* were chosen for further development. Table 1 shows part of another type of visualization, which ranks operations by how often they appear in minimal operation sets. This helped to determine the order in which operations were introduced to users in training materials. This visualization can also be linked back to the coded scenario data, allowing designers to see which

approaches led the optimization process to each data point.

## Conclusions and Future Work

Interface optimization can help designers weigh speed, simplicity, and versatility tradeoffs early in a UI design process. Further work is needed to make the technique practical. First, the process of coding scenarios would be easier with the help of some dedicated accounting tools. Second, faster optimization methods are needed: the current implementation uses exhaustive search and takes hours to complete. I hope to fix these problems and use interface optimization in a commercial project.

## References

1. Colwell, B., Davis, R.C., and Landay, J.A. *A study of early stage game design and prototyping*. University of Washington, Seattle, WA, 2008.
2. Davis, R.C., Colwell, B., and Landay, J.A. K-Sketch: a “kinetic” sketch pad for novice animators. *ACM* (2008), 413–422.
3. Johnson, J. and Henderson, A. Conceptual models: begin by designing what to design. *interactions* 9, (2002), 25–32.
4. Taatgen, N.A., Huss, D., Dickison, D., and Anderson, J.R. The acquisition of robust and flexible cognitive skills. *Journal of Experimental Psychology: General* 137, 3 (2008), 548–565.