

# The Role of Design Spaces

Mary Shaw, Carnegie Mellon University

// A case study involving a traffic signal simulator illustrates the benefits of considering the design space during design and the risks of failing to do so. //



A **CENTRAL TASK** in design is choosing what artifact will best satisfy the client's needs, whether that will require creating an artifact or choosing from existing alternatives. A design space identifies and organizes the decisions to be made, together with the alternatives for those decisions, thereby providing guidance for creating artifacts or a framework for comparing them.

Here, I discuss design spaces and present an example design space for a traffic signal simulation task. I show how this space enables comparison of the designs, and discuss the benefits of explicitly considering the design space during design—and the risks of failing to do so.

## Design Spaces

The *design space* for a problem is the set of decisions to be made about the

designed artifact together with the alternative choices for these decisions. A *representation of a design space* is one of the static textual or graphical forms in which a particular design space—or a subset of that space—may be rendered.

Intuitively, a design space is a discrete Cartesian space in which design decisions are the dimensions, possible alternatives are values on those dimensions, and complete designs are points in the space.

In this view, the design space is concrete. This contrasts with a common usage in which “design space” loosely refers to domain knowledge about the problem or even to all of a design activity's decisions, whether they regard problem analysis, the designed artifact, or the process of producing the design.

In practice, most interesting design spaces are too rich to represent in their entirety, so design space representations

feature dimensions corresponding to the properties of principal interest. Design dimensions aren't independent, so choosing an alternative for one decision might preclude alternatives for other decisions or make them irrelevant. For example, if displaying a value is optional, then decisions about the display format are irrelevant if the value isn't displayed. If a graph displays multiple values, all should use the same units. So, representing portions of the design space as trees is convenient, despite the disadvantage of implying an order in which to make decisions.

A design space's representation for a particular task is usually a slice of the complete design space that captures the important properties the artifact must have. By organizing design decisions, a design space helps designers consider relevant alternatives systematically. It also provides a way to compare similar products by highlighting differences between designs and allowing systematic matching to the needs of the problem at hand. Naturally, a good representation for a particular problem should reflect the solution's desired properties.

Design spaces can inoculate designers against the temptation to use the first alternative that comes to mind. For example, in studying software architectures, I repeatedly observed developers' tendency to use a familiar system structure instead of analyzing the problem to select an appropriate structure. Evidently, they were often oblivious even to the existence of alternatives; that is, they were defaulting to familiar structures instead of designing suitable ones.

Since at least 1971,<sup>1</sup> computer science has used design spaces to organize knowledge about families of designs or systems to describe

- computer architecture,<sup>1,2</sup>
- user input devices,<sup>3</sup>

- user interface implementation structures,<sup>4</sup>
- software architectural styles,<sup>5</sup>
- distributed sensors,<sup>6</sup> and
- typeface design.<sup>7</sup>

Design studies use the exploration of design spaces to find suitable designs, often by searching, as a model of designer action.<sup>8</sup>

Often—and in most practical problems of realistic size—the design space is not completely known in advance. In this case, the elaboration of the space proceeds hand-in-hand with the design process. Herbert Simon treated the task of selection from a fixed space as enumeration and optimization, and the task of searching an unknown or open-ended space as search and satisficing.<sup>9</sup> These cases align (roughly) with routine and innovative design.

## Representing Design Spaces

Figure 1 shows a small design space for sharing information via the Web. This is only a small slice of the entire design space, selected to compare representations of the same space. For this small example, each of the three dimensions has two possible values:

- *Activation*. Does the sender push the information to the reader, or does the reader pull the communication?
- *Privacy*. Is the communication private to a small set of known parties, or is it public?
- *Authorship*. Does one person or an open-ended group author the information?

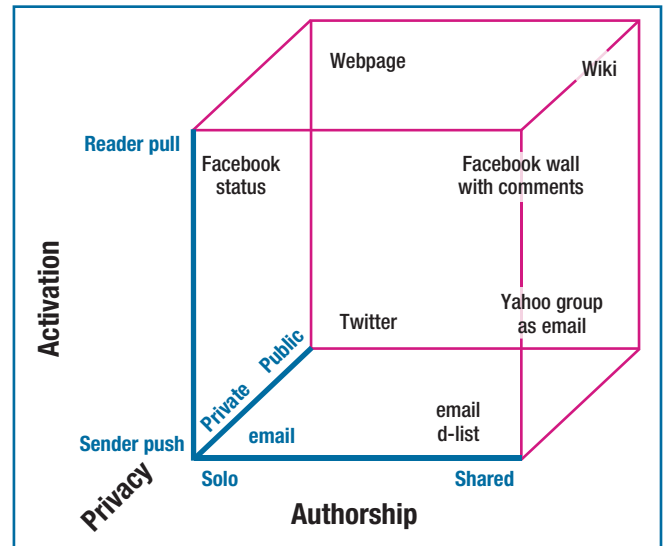
Figure 1 shows examples at every point of the space. For instance, a sender pushes email to the reader's mailbox, the sender writes it, and it's private

to the sender and named recipients. Of course, more than one application can occupy a point in this space. For example, instant messaging also lies at the <sender push, private, solo> point. Some points might also be unoccupied. This might happen because the combinations of choices don't make sense, or it might indicate an opportunity for new products.

Sketching multidimensional spaces obviously doesn't scale well. This small design space admits other representations. For example, in Table 1, rows correspond to points in the space, and columns correspond to the dimensions. This table's shortcoming is that it represents the points on each dimension only implicitly, in the values in the table's body. This format can represent the design space only to the extent that it's populated with a full range of examples.

You can also represent this design space by focusing on dimensions and their values. Figure 2 shows one such form. Following Frederick Brooks, the tree has two kinds of branches: choice and substructure.<sup>10</sup> Choice branches, flagged with “##,” are the actual design decisions; usually one option should be chosen. Substructure branches (not flagged) group independent decisions about the design; usually all of these should be explored.

Figure 2 has only one hierarchical level, but the format allows deeper structure; indeed, the traffic signal simulation space we'll look at is much



**FIGURE 1.** A small design space for Web information sharing. This representation selects three decisions about information sharing and shows how they correspond to some common applications.

richer. This representation's advantages are that it shows alternatives without relying on examples (Figure 2 adds two alternatives—interactive and login controlled) and it handles hierarchical descriptions well. Its disadvantage is that it represents a point in the space diffusely by tagging all relevant values. The figure illustrates this by placing the email, wiki, and (static) webpage instances in the representation.

Naturally, if other properties dominate design concerns, a different design space would be appropriate. The example I just discussed addresses how information flows between users. If the properties of interest are related to content representation and storage, the dimensions of interest might be <peristence, locus of state, latency, content type>.

## The Traffic Signal Simulation Design Space

To relate design spaces more closely to practice, I turn to a problem of a more realistic size, drawn from the US National Science Foundation-sponsored Studying Professional Software

TABLE 1

An instance-oriented representation of the design space in Figure 1. This alternative representation is organized around the points in the design space.

Instance	Activation	Privacy	Authorship
Webpage	Reader pull	Public	Solo
Wiki	Reader pull	Public	Shared
Facebook status	Reader pull	Private	Solo
Facebook wall with comments	Reader pull	Private	Shared
Twitter	Sender push	Public	Solo
Yahoo group as email	Sender push	Public	Shared
Email	Sender push	Private	Solo
Email d-list	Sender push	Private	Shared

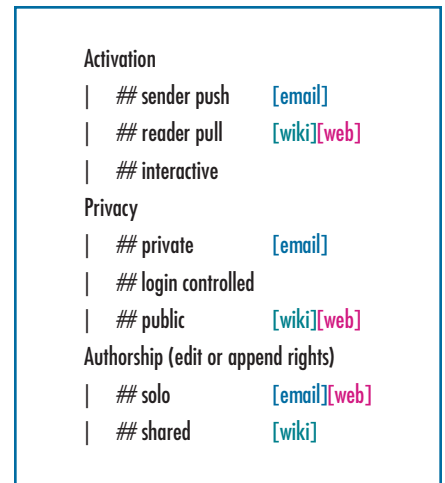


FIGURE 2. A dimension-oriented representation of the design space in Figure 1. A hierarchical representation of the design space of Figure 1 scales better to more dimensions.

Design workshop ([www.ics.uci.edu/design-workshop](http://www.ics.uci.edu/design-workshop)). In preparation for this workshop, organizers videotaped three two-person teams as they worked for one to two hours on a design problem. A multidisciplinary group of design researchers analyzed the sessions' tapes and transcripts, then discussed the sessions at the NSF workshop. The design task was a simulator for traffic flow in a street network, whose purpose was to help civil engineering students appreciate the subtlety of traffic-light timing. The full text of the task statement (the *prompt*) is at [www.ics.uci.edu/design-workshop/files/UCI\\_Design\\_Workshop\\_Prompt.pdf](http://www.ics.uci.edu/design-workshop/files/UCI_Design_Workshop_Prompt.pdf) and in the introduction to a special issue of *Design Studies* that reports some of the results.<sup>11</sup>

Figure 3 presents a representation of the design space implicit in the transcripts. None of the teams explicitly considered a design space, so to develop Figure 3, I studied the videos and transcripts and identified the principal conceptual entities the teams included in their designs along with any alternatives they considered. Trying to remain faithful to the structure that emerged from the design discussions, I identified seven principal dimensions to organize the alternatives and annotated them

with details from the discussions. The dimensions are

- System Concept,
- Road System,
- Traffic Signals,
- Traffic Model,
- Simulator,
- Model of Time, and
- User Interface.

The elaboration of each of these dimensions is hierarchical. In Figure 3, AD, IN, and MB indicate the three teams' decisions.

I also reviewed the prompt, noting choices that the prompt itself implied. In some cases, I added obvious alternatives that did not otherwise appear in the transcripts. The prompt clearly implies certain design decisions, but the team choices were often strikingly different from those decisions and from other teams' decisions. For example, the prompt says, "Students must be able to describe the behavior of the traffic lights at each of the intersections," which quite clearly indicates that students should set and vary the signals' timing. Finally, I examined the demonstration version of Trafficware ([www.trafficware.com](http://www.trafficware.com)), a commercial traffic simulation tool. This professional tool

has obviously received more design and development effort than the workshop exercise. Nevertheless, it's informative to see where it lies in the design space. In Figure 3, the bold red boxed text flags the prompt's implications while highlighted backgrounds indicate Trafficware's decisions.

The resulting representation of the design space is incomplete in two important ways. First, the alternatives don't exhaust the possibilities. Indeed, Trafficware presents many more possibilities. Second, this representation captures only the larger-grained and (apparently) most significant design decisions. Omitted, for example, are the characterization of traffic entering and leaving at the map edges, the handling of left turns, and analytics. Nevertheless, Figure 3's representation provides a uniform framework for comparing the designs the workshop studied.

### Through a Design Space Lens

Figure 3 provides a basis for comparing the three teams' diverse approaches and reflecting on ways that explicit consideration of the design space might have

helped them. I'll concentrate on architectural decisions, which fall chiefly under the System Concept and Simulator dimensions.

System Concept corresponds to the choice of the overall system architecture and thereby provides the structure for the rest of the design. (By "architecture," I mean the high-level concepts that guide system organization, not the selection of data structures or the class structure of an object-oriented system. Thus, "we'll use MVC [model-view-controller]" is architectural, but "a road is a queue" isn't.) Perhaps because of the workshop's short time frame, the teams spent little time explicitly discussing overall organization. Each picked a different system concept. In each case, the team identified the top-level organization almost automatically, without considering and evaluating alternatives. It appears from the transcripts that the teams chose the overall system concept implicitly, more as a default than a deliberate decision.

Team AD couched their discussion in terms of objects. They first selected data structures for intersections, roads, and the "cop," which was the main object to advance the state of the system via timer events. A high-level network object let users add roads, and the simulation would infer the intersections. The team mentioned the MVC pattern, but instead of using it to organize the design activity, they periodically tried to decide whether some entity (the cop or the clock) was a model or controller.

Team IN saw a data-driven problem in the prompt, so they focused on data items. Although they also mentioned MVC, they centered the design on a main map, to which the user added intersections connected to roads. The main map was the overall controller; intersections were also active objects that query roads for traffic and enforce safety rules on lights.

Team MB focused on the problem's visual aspects and considered things

System Concept			Traffic Model		
##MVC	AD		## Master traffic object, discrete cars	MB	
##Code + User interface	IN		## <b>Discrete cars</b>		
## User Interface	MB		## Cars with state, route, destination	MB	
## <b>Simulator</b>			## Random choices at intersections	AD IN MB	
			## Distributions only		
<b>Road System</b>			<b>Simulator</b>		
High-level organization			## MVC		
## Intersections	AD		## Set of objects		
## Roads			## executing in parallel threads	IN MB	
## <b>Network</b>	AD IN		## traversed by a controller		
Intersections			at each clock tick	AD	
## <b>Collection of signals</b>	IN		## <b>Separate model and simulation engine</b>		
## Signals and sensors in approaches	MB				
## Have roads (with lights and cars)	AD				
Roads			<b>Model of Time</b>		
Lanes			## <b>Uniform time ticks</b>	AD	
## No lanes			## Scheduled event queue		
## <b>Lanes, with signal per lane</b>	AD IN		## Parallel threads	IN	
Throughput					
<b>Capacity</b>	AD		<b>User Interface</b>		
<b>Latency</b>	IN MB		Display		
Connection of roads to intersections			Layout of visual map		
## Intersections have queues (roads)	AD		## <b>student's own choosing</b>		
## Lights and sensors in approaches	MB		## intersections implied by		
## Unspecified or unclear	IN		road crossings	AD MB	
## <b>Simulator handles interaction</b>			Relation of layout distances to road length		
			## layout determines road length	MB	
<b>Traffic Signals</b>			## layout determines length,		
Place in hierarchy			constrained to grid	AD	
## Belong to roads	AD		## <b>length independent of layout</b>		
## <b>Belong to intersections</b>	IN		Defining the map		
## Belong to approaches,			## <b>click-drag-drop visual editing</b>	AD IN MB	
which connect roads to ints	MB		<b>Setting light timing</b>		
			## <b>double-click on intersection</b>	AD MB	
<b>Safety</b>			Defining traffic model		
## Independent lights with safety checks			## <b>set traffic loads only at edges</b>	AD IN MB	
## Controller checks dynamically	AD IN		## allow traffic to enter internally		
## UI checks at definition time	MB		Viewing results		
## <b>One set per intersection, selected from safe set</b>			## <b>see individual cars, lights</b>	MB	
Relations among intersections			## see view of density on roads	MB	
## Independent	AD		## see cars and aggregate statistics		
## <b>Synchronized</b>	IN MB		## see aggregate statistics only		
Setting timing			Saving and restoring		
## System sets timing	AD IN MB		## <b>supported</b>	MB	
## <b>Students set timing</b>	MB		## not supported	AD	
Sensors					
## Immediately advance on arrival	IN				
## <b>Wait to synchronize</b>					

FIGURE 3. A composite representation of several designs in the traffic signal simulation design space, showing the decisions implied by the task statement (boxed red text), made by the three teams (AD, IN, and MB), and made by a commercial product (highlighted in yellow).

the students must do: build the map, create traffic patterns, set signal timings, run the simulation, and so on. They organized the design around a drawing tool to support these activities,

and simulation was one of the invocable actions.

Both AD and IN used MVC informally. In classic MVC, the controller is chiefly a dynamic mediator between

## ABOUT THE AUTHOR



**MARY SHAW** is the Alan J. Perlis University Professor of Computer Science at Carnegie Mellon University. Her research interests include software design, software architecture, end-user software engineering, and cybersociotechnical systems. Shaw has a PhD in computer science from Carnegie Mellon. She's a fellow of the ACM, the IEEE, and the American Association for the Advancement of Science; she's also a member of International Federation for Information Processing Working Group 2.10 on Software Architecture. Contact her at [mary.shaw@cs.cmu.edu](mailto:mary.shaw@cs.cmu.edu).

the user's actions (through the view of the user interface) and the domain logic that is captured in the model. All three designs, especially AD, assigned the controller the details of running the simulation. When the simulation is running, however, the user isn't offering input to the system. So, incorporating the simulation logic in the controller might make sense for the common informal meaning of "controller," but it's not a good match for the MVC pattern's controller.

Figure 3's framework helps identify the differences among the system concepts and simulation mechanisms in these three designs. It also highlights the core task set by the first sentence of the prompt—"designing a traffic flow simulation program"—and the later charge to "focus on the important design decisions that form the foundation of the implementation."

A simulator is a well-known type of software system with a history that goes back many decades. Identifying a system as a simulator leads to recognizing—and separating in the design—four concerns:

- the model of the phenomenon to simulate,
- the means of setting up a specific case to simulate,
- the simulation engine itself, and
- a simulation's current state (including a way to report results).

MB focused on the problem's simulation aspect, although they didn't say much about the simulation engine.


Recognizing the conventional simulator structure might have helped AD separate "control"—that is, running a simulation on a specific case—from the MVC controller (which would mediate between the user interface and the data the students defined). Viewing the system as a simulator might have led AD and IN to consider alternatives to massively parallel execution of objects. It would have almost certainly helped MB separate the user interface featuring a drawing tool from the model that the drawing tool was to create.

Organizing design knowledge as a design space provides a framework for systematically considering design alternatives, for recognizing interactions and trade-offs among decisions, and for comparing designs.

Had a design space been available for the traffic signal simulation task, it would have provided a checklist of questions to consider along with possible alternatives. Even if the design space hadn't been available at the outset, the discipline of creating a partial representation would have sensitized the designers to the existence of alternatives and helped organize the design discussion.

Indeed, incorporating design spaces into normal practice would lead designers to ask whether a design space had already been developed for this or a similar problem. This would help avoid problem analysis from scratch and help designers exploit domain expertise encoded in the design space.

A suitable representation of a design

space also supports comparison of designs—in particular, the selection of an appropriate solution from a set of existing alternatives that have been identified with points in the space. If the requirement is mapped to one or more points in the space, the solutions at nearby points in the space should be favored candidates. 

### Acknowledgments

US National Science Foundation grant CCF-0845840 partially supported the Studying Professional Software Design workshop. The workshop wouldn't have been possible without the professional designers' willingness to work on the design task and to allow review of that work.

### References

1. C.G. Bell and A. Newell, *Computer Structures: Readings and Examples*, McGraw-Hill, 1971.
2. D. Sima, "The Design Space of Register Renaming Techniques," *IEEE Micro*, vol. 20, no. 5, 2000, pp. 70–83; doi:10.1109/40.877952.
3. S.K. Card, J.D. Mackinlay, and G.G. Robertson, "A Morphological Analysis of the Design Space of Input Devices," *ACM Trans. Information Systems*, vol. 9, no. 2, 1991, pp. 99–122.
4. T.G. Lane, *User Interface Software Structures*, doctoral dissertation, School of Computer Science, Carnegie Mellon Univ., 1990.
5. M. Shaw and P. Clements, "A Field Guide to Boxology: Preliminary Classification of Architectural Styles for Software Systems," *Proc. 21st Int'l Computer Software and Applications Conf.*, IEEE CS Press, 1997, pp. 6–13.
6. K. Römer and F. Mattern, "The Design Space of Wireless Sensor Networks," *IEEE Wireless Comm.*, vol. 11, no. 6, 2004, pp. 54–61.
7. "The Design Space," TypEdu; [www.typedu.org/dynamic/lessons/article/designspace](http://www.typedu.org/dynamic/lessons/article/designspace).
8. R.F. Woodbury and A.L. Burrow, "Whither Design Space?" *Artificial Intelligence for Eng. Design, Analysis, and Manufacturing*, vol. 20, no. 2, 2006, pp. 63–82.
9. H.A. Simon, *Sciences of the Artificial*, MIT Press, 1996, ch. 5.
10. F.P. Brooks Jr., *The Design of Design: Essays from a Computer Scientist*, Addison-Wesley, 2010.
11. M. Petre, A. van der Hoek, and A. Baker, "Editorial," *Design Studies*, vol. 31, no. 6, 2010, pp. 533–544.



Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.