# The Role of Design Spaces in Software Design

**Mary Shaw**
Institute for Software Research
School of Computer Science
Carnegie Mellon University
Pittsburgh PA 15213
+1 412-268-2589
mary.shaw@cs.cmu.edu

## ABSTRACT

A central task in design is deciding what artifact will best satisfy the client's needs, whether by creating a new artifact or choosing from among existing alternatives. A design space identifies and organizes the decisions to be made, together with the alternatives for those decisions, thereby providing guidance for creation or a framework for comparison. The workshop *Studying Professional Software Design* studied three pairs of professional software designers sketching designs for a traffic signal simulator. This paper presents a representation of the design space for the traffic signal simulation task. It shows how this design space enables comparison of the designs, and it discusses the benefits of explicitly considering the design space during design and the risks of failing to do so.

**Categories and Subject Descriptors**

D.2.10 [**Software Engineering**]: Design (also .b and .c)

**General Terms**

Design

**Keywords**

software design, design space

## 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 very concrete. This is in contrast to a common, much vaguer, usage in which "design space" refers loosely to domain knowledge about the problem or perhaps to all decisions in a design activity, be they about the problem analysis, designed artifact, or the process of producing the design.

In practice, most interesting design spaces are too rich to represent in their entirety, so representations of a design space select dimensions corresponding to the properties of principal interest. Design dimensions are not independent, so choosing some alternative for one decision may 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 is not displayed; if multiple values are to be displayed as a graph, all should use the same units. As a result, it is convenient to represent portions of the design space as trees, despite the disadvantage of implying an order in which decisions should be made.

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

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

Design spaces have been used in computer science to organize knowledge about families of designs or systems since at least 1971 [BN71]. They have been used, among other things, to describe computer architecture [BN71, Si00], user input devices [CMR91], user interface implementation structures [La90], software architectural styles [SC97], distributed sensors [RM04], and typeface design [Type]. Exploration of design spaces to find suitable designs, often by searching, is used as a model of designer action in design studies [WB06].

Often – and in most practical problems at scale – the design space is not completely known in advance. In these cases the elaboration of the space proceeds hand-in-hand with the design process. Simon [Si96] addresses the difference between the two cases in Chapter 5, treating 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. These cases align (very roughly) with routine and innovative design.

## REPRESENTING DESIGN SPACES

Figure 1 shows a small design space for information sharing via the world-wide web. This is only a small slice of the entire design space, selected to compare representations of the same space. The three dimensions, each with two possible values for this small example, are

- *Activation:* Is communication driven by the sender pushing the information to the reader or by the reader pulling the communication?
- *Privacy:* Is the communication private to a small set of known parties, or is it public?
- *Authorship:* Is the information authored by a single person or by an open-ended group?

Figure 1 shows examples at each point of the space. For example, email is pushed by the sender to the mailbox of the reader, it is authored by the sender, and it is private to the sender and named recipients.

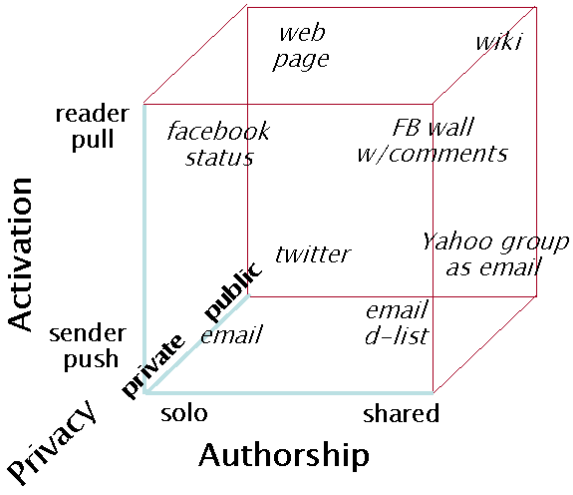


**Figure 1: Design space for WWW information sharing**

Of course, points in this space may be occupied by more than one application. For example, instant messaging also lies at the <push, private, solo> point. It is also possible for some points to be unoccupied, as in Figure 3. This might happen because the combinations of choices don't make sense, or it might indicate opportunity for new products.

Sketching multi-dimensional spaces obviously does not scale well. This small space can be represented in other ways. Figure 2 uses a tabular form with rows corresponding to points in the space and columns to the dimensions; it has the shortcoming that the points on each dimension are represented only implicitly, in the values in the body of the table. The design space can be represented in this format only to the extent that it is populated with a full range of examples.

| Instance | Activation | Privacy | Authorship |
|---|---|---|---|
| web page | reader pull | public | solo |
| wiki | reader pull | public | shared |
| facebook status | reader pull | private | solo |
| facebook wall w/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: Instance-oriented representation of design space of Figure 1**

This design space can also be represented with emphasis on the dimensions and their values. Figure 3 shows one such form. Following Brooks [Br10], the tree has two kinds of branches: choice and substructure. 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.

Activation
| ## sender push **[email]**
| ## reader pull **[wiki] [web]**
| ## interactive
Privacy
| ## private **[email]**
| ## login controlled
| ## public **[wiki] [web]**
Authorship (edit or append rights)
| ## solo **[email] [web]**
| ## shared **[wiki]**

**Figure 3: Dimension-oriented representation of design space of Figure 1**

This example has only one hierarchical level, but the format admits of deeper structure, and indeed the traffic signal simulation space is much richer. This representation has the advantage of showing alternatives without relying on examples (two new alternatives are added here), and it handles hierarchical descriptions well. Its disadvantage is that a point in the space is represented diffusely, by tagging all relevant values. This is illustrated here by placing email, wiki, and (static) web page in this representation.

Naturally, if other properties are of interest, a different design space would be appropriate. For example, this example addresses the way information flows between users. If the properties of interest were about the representation and storage of content, the dimensions of interest might be <Persistence, Locus of State, Latency, Content Type>.

## A DESIGN SPACE FOR TRAFFIC SIGNAL SIMULATION

To relate design spaces more closely to practice, I turn to a problem of more realistic size, drawn from the NSF-sponsored workshop *Studying Professional Software Design* [SPSD10]. In preparation for this workshop, three 2-person teams were videotaped as they worked for 1-2 hours on a design problem. The tapes and transcripts of these sessions were analyzed by a multi-disciplinary group of design researchers, who then met to discuss the sessions. The design task was a simulator of traffic flow in a street network, to be used by civil engineering students to appreciate the subtlety of traffic light timing. The full text of the task statement (the "prompt") is given in the web site of the workshop [SPSD10a] and the introduction to a special issue of *Design Studies* [PvB10] that reports some of the workshop results.

Figure 4 presents a representation of the design space implicit in the transcripts. None of the teams explicitly considered a design space, so to develop Figure 4 I studied the videos and transcripts of the design sessions and identified the principal conceptual entities the teams included in their designs, along with any alternatives they considered. Trying to be faithful to the structure that emerged from the design discussions, I identified a set of seven principal dimensions to organize the alternatives and annotated these with details from the discussions. These are

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

The elaboration of each of these dimensions is hierarchical.

I also reviewed the task prompt, noting choices that were implied by the prompt itself. In some cases I added obvious alternatives that did not otherwise appear in the transcripts. Finally, I examined the demonstration version of a commercial traffic simulation tool [Traf11]. This is a professional tool, and it has obviously received more design and development effort than the workshop exercise,. Nevertheless, it is informative to see where it lies in the design space.

Figure 4 show the decisions made by these three teams, denoted **AD**, **IN**, and **MB**. The diversity among the three designs is striking. The prompt clearly implies certain design decisions. For example, it 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 timing of the signals. These implications of the prompt are flagged in Figure 4 with **bold red boxed text**. Finally, the decisions exhibited by the commercial tool are indicated in Figure 4 with highlighted backgrounds.

The resulting representation of the design space is incomplete in two important ways. First, the alternatives do not exhaust the possibilities. Indeed, the commercial traffic simulation product [Tra11] presents many such 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 edges of the map, the handling of left turns, and analytics. Nevertheless, the representation of Figure 4 provides a uniform framework for comparing the designs studied by the workshop.

## VIEWING DESIGNS THROUGH THE LENS OF A DESIGN SPACE

Figure 4 provides a basis for comparing the approaches of the three design teams and reflecting on ways that explicit consideration of the design space might have helped the designers. I will concentrate on architectural decisions, which fall chiefly in the "System Concept" and "Simulator" dimensions.

The "System Concept" dimension corresponds to the choice of the overall system architecture[1] and thereby provides the structure for the rest of the design. Perhaps because of the short time frame imposed by the workshop setting, the teams spent little time explicitly discussing overall organization. Each of the three teams 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 overall system concept was chosen implicitly, more as a default than a deliberate decision.

Team AD couched their discussion in terms of objects. They began by selecting 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 allowed users to add roads, from which intersections were inferred. The MVC pattern was mentioned, but instead of using the model-view-controller pattern to

[1] By "architecture" I mean the high-level concepts that guide the system organization, not the selection of data structures or the class structure of an object-oriented system. Thus "we'll use MVC" is architectural, but "a road is a queue" is not.

organize the design activity, they periodically tried to decide whether an entity (the cop, the clock) was a model or controller.

Team IN saw a data-driven problem in the prompt, so they focused on the data items. Although they also mentioned MVC, they centered the design on a main map, to which the user added intersections connected to roads. This 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 visual aspect of the problem, and their discussion considered things the students must do: build the map, create traffic patterns, set signal timings, run the simulation, and so forth. They organized the design around a drawing tool to support these activities, and simulation was one of the invokable actions.

Having mentioned MVC, both AD and IN used the pattern informally, and neither mentioned MVC in their summary presentation. In classic MVC, the controller is chiefly a dynamic mediator between the user's actions (through the UI) and the domain logic in the model. In these designs, especially for AD, the controller was assigned the details of running the simulation. When the simulation is running, however, the user is not offering input to the system. Thus, incorporating the simulation logic in the controller may make sense for the common informal meaning of "controller", but it's not a good match for the controller of the MVC pattern.

The framework of Figure 4 helps us to identify these differences among the system concepts and simulation mechanisms chosen by the teams. 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 be simulated,
- the means of setting up a specific case to simulate,
- the simulation engine itself, and
- the current state of a simulation (including a way to report results).

Team MB focused on the simulation aspect of the problem, though they did not say much about the simulation engine. Recognizing the structure of a simulator might have helped AD separate "control" (i.e., running a simulation on a specific case) from the MVC controller (which would mediate between the UI and the data being defined by the student). 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 UI featuring a drawing tool from the model that was being created with the drawing tool.

## IMPLICATIONS FOR PRACTICE

Organizing design knowledge as a design space provides a framework for systematically considering design alternatives, for recognizing interactions and tradeoffs 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 and possible alternatives. Even if the design space had not been available at the outset, the discipline of creating a partial representation would have sensitize the designer to the existence of alternatives and helped to organize the design discussion.

Indeed, incorporating the use of design spaces in normal practice would lead designers to ask whether a design space had already been developed for this or a similar problem, to avoid problem analysis from scratch and to 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.

## ACKNOWLEDGEMENTS

The workshop on Studying Professional Software Design was partially supported by NSF grant CCF-0845840. The workshop would not have been possible without the willingness of the professional designers to work on the design task and to allow that work to be reviewed.

**System Concept**

| | |
|---|---|
| ##MVC | AD |
| ##Code + User interface | IN |
| ## User Interface | MB |
| ## Simulator | |

**Road System**

| | |
|---|---|
| High-level organization | |
| ## Intersections | AD |
| ## Roads | |
| ## Network | AD IN |
| Intersections | |
| ##Collection of signals | IN |
| ## Signals and sensors in approaches | MB |
| ## Have roads (with lights and cars) | AD |
| Roads | |
| Lanes | |
| ## No lanes | |
| ## Lanes, with signal per lane | AD IN |
| Throughput | |
| Capacity | AD |
| Latency | IN MB |
| Connection of roads to intersections | |
| ## Intersections have queues (roads) | AD |
| ## Lights and sensors in approaches | MB |
| ## Unspecified or unclear | IN |
| ## Simulator handles interaction | |

**Traffic Signals**

| | |
|---|---|
| Place in hierarchy | |
| ## Belong to roads | AD |
| ## Belong to intersections | IN |
| ## Belong to approaches, which connect rds to ints | MB |
| Safety | |
| ## Independent lights with safety checks | |
| ##Controller checks dynamically | AD IN |
| ## UI checks at definition time | MB |
| ## One set per intersection, selected from safe set | |
| Relations among intersections | |
| ## Independent | AD |
| ## Synchronized | IN MB |
| Setting timing | |
| ## System sets timing | AD IN MB |
| ## Students set timing | MB |
| Sensors | |
| ## Immediately advance on arrival | IN |
| ## Wait to synchronize | |

**Traffic Model**

| | |
|---|---|
| ## Master traffic object, discrete cars | MB |
| ## Discrete cars | |
| ## Cars with state, route, destination | MB |
| ## Random choices at intersections | AD IN MB |
| ## Distributions only | |

**Simulator**

| | |
|---|---|
| ## MVC | |
| ## Set of objects | |
| ## executing in parallel threads | IN MB |
| ## traversed by a controller at each clock tick | AD |
| ## Separate model and simulation engine | |

**Model of Time**

| | |
|---|---|
| ## Uniform time ticks | AD |
| ## Scheduled event queue | |
| ## Parallel threads | IN |

**User Interface**

| | |
|---|---|
| Display | |
| Layout of visual map | |
| ## student's own choosing | |
| ## intersections implied by road crossings | AD MB |
| Relation of layout distances to road length | |
| ## layout determines road length | MB |
| ## layout determines length, constrained to grid | AD |
| ## length independent of layout | |
| Defining the map | |
| ##click-drag-drop visual editing | AD IN MB |
| Setting light timing | |
| ## double-click on intersection | AD MB |
| Defining traffic model | |
| ## set traffic loads only at edges | AD IN MB |
| ## allow traffic to enter internally | |
| Viewing results | |
| ## see individual cars, lights | MB |
| ## see view of density on roads | MB |
| ## see cars and aggregate statistics | |
| ## see aggregate statistics only | |
| Saving and restoring | |
| ## supported | MB |
| ## not supported | AD |

**Figure 4: A composite representation of several designs in the traffic signal simulation design space, showing the decisions implied by the prompt, the three teams' design decisions AD IN MB, and a commercial product**

## REFERENCES


[BN71] C. Gordon Bell and Allen Newell. *Computer Structures: Readings and Examples.* McGraw-Hill 1971.

[Br10] Frederick P. Brooks, Jr. *Design of Design*. Addison-Wesley 2010.

[CMR91] Stuart K. Card, Jock D. Mackinlay, and George G. Robertson. A morphological analysis of the design space of input devices. *ACM Transactions on Information Systems*. 9 (2) April 1991, 99-122.

[La90] Thomas G. Lane. *User Interface Software Structures*. PhD thesis, Carnegie Mellon University, May 1990.

[PvB10] Marian Petre, André van der Hoek, and Alex Baker. Editorial (introduction). Special Issue Studying Professional Software Design, *Design Studies*, vol 31, no 6, Nov 2010, 533-544.

[RM04] Kay Römer and Friedemann Mattern. The design space of wireless sensor networks. *IEEE Wireless Comm*. Dec 2004.

[SC97] Mary Shaw and Paul Clements. A field guide to Boxology: Preliminary classification of architectural styles for software systems, *Proceedings of the 21st International Computer Software and Applications Conference*, 1997, pp. 6-13.

[Si96] Herbert A. Simon. Sciences of the Artificial. MIT Press, 3rd edition 1996.

[Si00] Dezsö Sima, The Design Space of Register Renaming Techniques, *IEEE Micro*, vol. 20, no. 5, pp. 70-83, Sep./Oct. 2000, doi:10.1109/40.877952

[SPSD10] International workshop "Studying Professional Software Design", February 8-10 2010. http://www.ics.uci.edu/design-workshop

[SPSD10a] *Design Prompt: Traffic Signal Simulator*. Problem statement for International workshop on Studying Professional Software Design. http://www.ics.uci.edu/design-workshop/files/UCI_Design_Workshop_Prompt.pdf

[Traf11] Trafficware. SynchroGreen Adaptive Traffic Control System. http://www.trafficware.com/

[WB06] Robert F. Woodbury and Andrew L. Burrow. Whither design space? *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, vol 20 no 2, 2006, pp. 63-82.

[Type] Design space for typeface design http://www.typedu.org/dynamic/lessons/article/designspace


## 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. She has received the ACM SIGSOFT Outstanding Research AWARD, the IEEE Computer Society TCSE's Distinguished Educator Award, CSEE&T's Nancy Mead Award for Excellence in Software Engineering Education, the Stevens Award, and the Warnier Prize. She is a fellow of the ACM, the IEEE, and the American Association for the Advancement of Science; she is also a member of IFIP WG 2.10 on Software Architecture. Contact her at mary.shaw@cs.cmu.edu