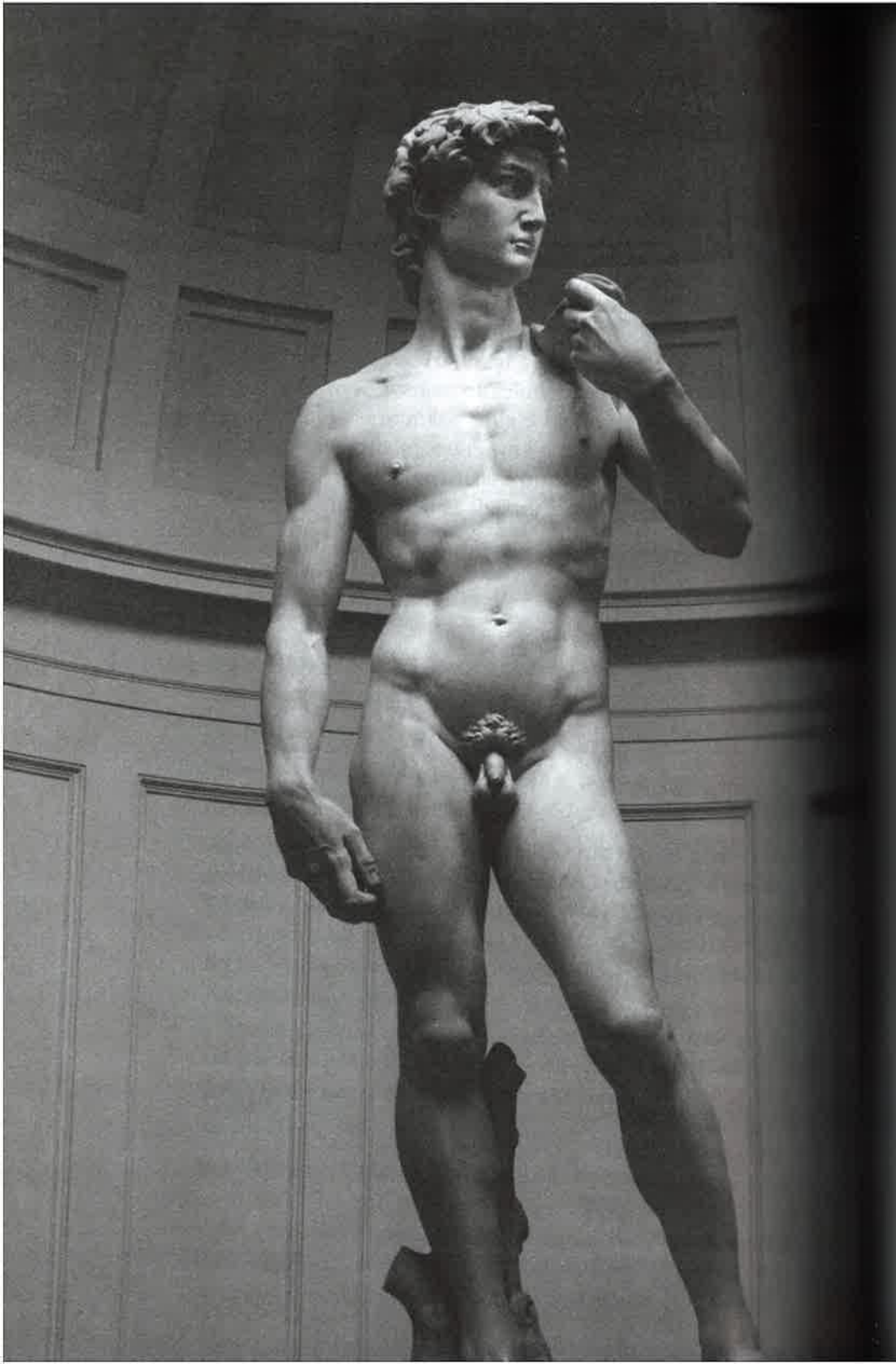

11
Constraints Are Friends

Form is liberating.

ARTISTS' APHORISM

I need four walls around me, to hold my life, and keep me from going astray.

JAMES TAYLOR, "BARTENDER'S BLUES"

A general-purpose product is harder to design well than a special-purpose one.

Michelangelo's *David*, carved from an "abandoned" block of marble
iStockphoto

Constraints

Constraints may be burdens, but they also may be friends. Constraints shrink the designer's search space. By so doing, they focus and speed design. Many of us disliked "Write an essay on whatever you want" assignments in junior high, and we were on to something real: removing all constraints makes the task of "designing" the essay harder, not easier.

Bach understood this. Wolff says, "Bach, predisposed from the very beginning toward traversing conventional boundaries, nevertheless preferred to work within a given framework and accept the challenges it imposed."[1]

Constraints not only shrink the search space, they challenge the designer, often thereby stimulating a completely fresh creation. Consider Michelangelo's *David*. According to legend, the block of marble had been abandoned by Antonio Rossellino 25 years earlier as unusable, because of a crack. The result is a *concept* of David different from that in previous and contemporary art. One is piqued to study exactly how Michelangelo coped with the block's defects, and how this stimulated the different artistic concept.

Christopher Wren's London churches offer another vivid example. Commissioned to rebuild 50 Anglican churches destroyed by the Great Fire of 1666, Wren was sorely constrained. For each, he had to take as given the site, its environment, and often the previous foundation. Moreover, Anglican churches must be oriented with the altar to the east in symbolic anticipation of Christ's promised return as the "bright morning star."[2] It is great fun to go today to each of the 27 Wren churches that remain after the attrition of time and the World War II Blitz. Look at each site and its problems, consider the orientation constraint, and see how Wren invented different solutions.

The designers of a viaduct on the Blue Ridge Parkway in the North Carolina mountains had to touch ground as little as possible to minimize environmental damage. The result was quite elegant.[3]

Up to a Point

Artificial constraints for one's design task have the nice property that one is free to relax them. Ideally, they push one into an unexplored corner of a design space, stimulating creativity. But any constraint set may push the designer into an empty corner, where no conceivable design works.

Therefore, one must carefully distinguish

- Real constraints
- Obsolete once-real constraints
- Constraints misperceived as real
- Intentional artificial constraints

Obsolete Constraints. The experienced designer, like a lion accustomed to pace the confines of its cage, may find himself obeying by habit constraints made obsolete by technological advances. Chapter 9, "Ten pounds in a five-pound sack," in *The Mythical Man-Month* [1975], now reads like a joke. It teaches techniques for squeezing software into cramped memory spaces. Crucial in 1965, they were already less important in 1975, but many programmers still strained for small sizes. (Of course, memory size has always mattered for embedded computers, especially now for those stunningly rich systems we still call "cell phones.")

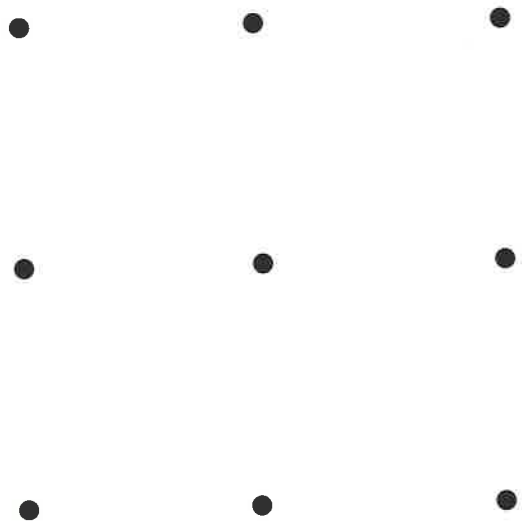

Figure 11-1 Misperceived constraint puzzle: Run a continuous line of no more than four straight segments through all nine dots.

Misperceived Constraints. These are more subtle. Figure 11-1 shows a classic example. (See Figure 11-5 for a solution.[4]) The boundary-line constraint described in Chapter 3 is another.

To multiply two 2 x 2 matrices in only seven multiplications instead of eight, one must discard the misperceived constraint that vector operations must be used.

The design of the IBM 9020 computer system for the Federal Aviation Administration (FAA) furnishes a painful example. System architects at MITRE Corporation, acting for the FAA, were properly aiming at a super-reliable system. They specified the configuration shown in Figure 11-2.

So far, so good. The IBM team responding to this request for bids discovered that the new System/360 semi-integrated

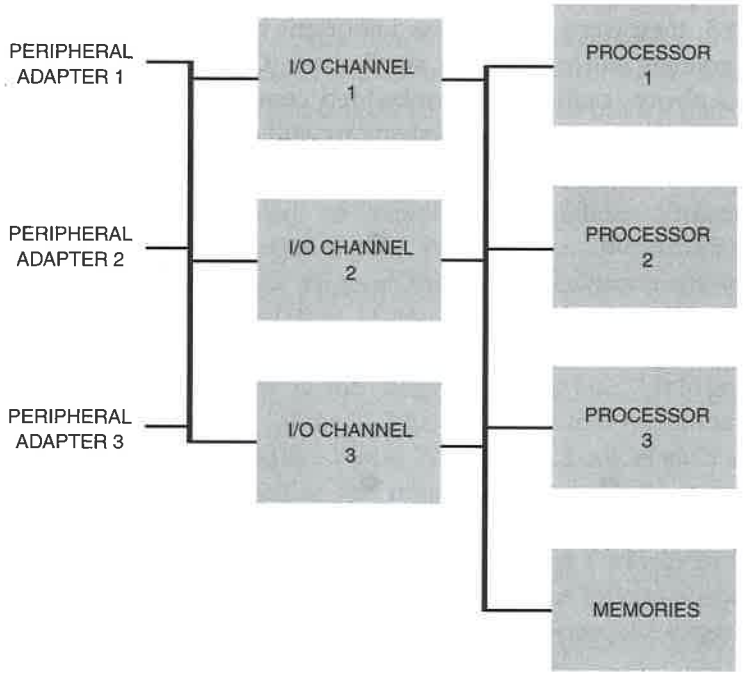

Figure 11-2 Triple modular redundant processor and I/O configuration specified for 1965 FAA system

circuit technology was very well suited for the unit-reliability demands.

The S/360 Model 50, a mid-range computer, more than met both the speed and reliability requirements for the processors. The same Model 50's I/O system, implemented with the same memory and datapaths as the processor, but with different microcoding, handsomely met the requirements for the I/O controller.

So the IBM engineers designed the system sketched in Figure 11-3.

Careful analysis showed that the Figure 11-3 configuration more than met all the system performance requirements and all the system reliability requirements. But it was rejected.

It did not have the specified configuration topology. The MITRE system architects had mistakenly insisted on the specified topology as an essential constraint—although what they were actually shopping for was function and reliability, not topology. So IBM bid, built, and delivered the configuration

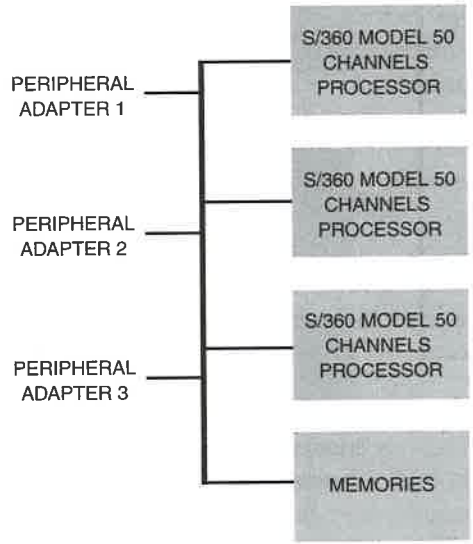

Figure 11-3 FAA system as first proposed with System/360 Model 50 computers

shown in Figure 11-4. Unchallenged reliability analyses showed this configuration to be in fact considerably *less* reliable than that in Figure 11-3, since there are twice as many components and many more connectors that can fail. But it met the specified constraints!

The pricing was magical for the taxpayers. Quite independently from cost, the government paid for system 11-4 just what it would have paid for system 11-3. IBM really wanted that contract! Of course, the lifetime costs for power, cooling, and maintenance would not have been at all equal.

When you specify something to be designed, tell what properties you need, not how they are to be achieved.

If implementation approaches are given as constraints, better solutions are cut off. For the sake of the artifact and the user, the designer confronted with false constraints should fight back!

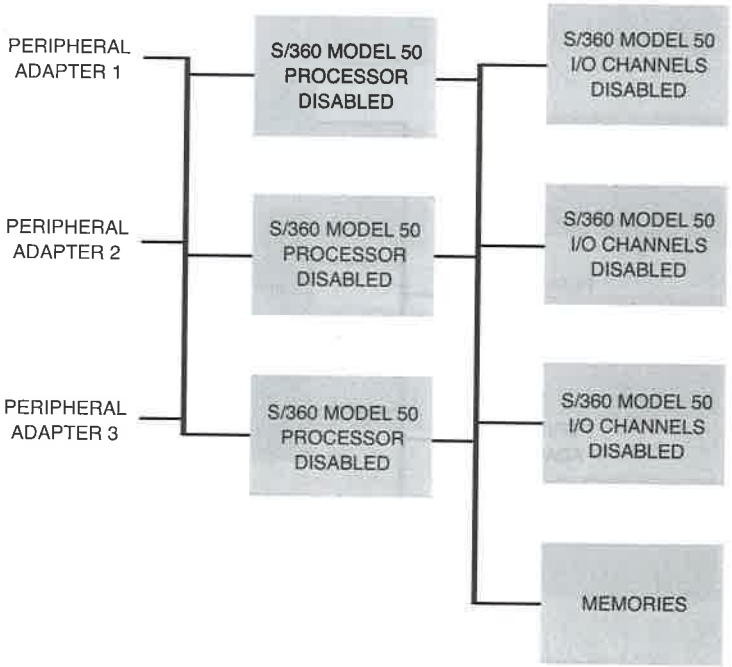

Figure 11-4 IBM 9020 FAA system as delivered

A Design Paradox: General-Purpose Artifacts Are Harder than Special-Purpose Ones

I have earlier argued that the hardest part of designing is deciding exactly *what to design*. Above, I argued that constraints are friends, that they make the design task easier, not harder. It follows that the more specialized the purpose, the easier the design task.

At first this seems wrong. One would offhand think it an easier assignment to "Design a 1,000-square-foot house" than to "Design a 1,000-square-foot house for a family with two children of opposite genders, located in Chapel Hill, North Carolina, facing north."

In one sense, of course, the former task is indeed easier—it is harder for the design to be criticized. If there are no constraints, there are no criteria for excellence. In general, it is easier to do a mediocre general-purpose design than to do a mediocre special-purpose design.

Nevertheless, overall the latter job is easier if one wants an excellent design. Any design process begins with the designer elaborating and particularizing the objectives and constraints. The first task is to narrow the design space. The more constrained the assigned goal, the more of this task has already been accomplished.

Dashing Off a General-Purpose Design. How can this be so? Consider the case of computer architecture. General-purpose computers are well understood. Over a hundred exemplar architectures have been built and sold. Everybody knows what they have to do. A good designer can sketch one out in a few days; the set of architectural decisions is clear:

- Instruction formats
- Addressing and memory management
- Datatypes and their representations
- Operation set
- Instruction sequencing
- Supervisory facilities
- Input-output

Designing the Special-Purpose Computer Architecture. On the other hand, designing a special-purpose computer clearly takes a lot of extra work up front. One must study the application. What makes it peculiar? What are the relative frequencies of operations? What are the weightings of desiderata for the clients: performance, cost, reliability, weight? Indeed, one has to develop an explicit characterization of the application.

Designing an Excellent General-Purpose Architecture. But one also needs an explicit user model to do proper design of a general-purpose architecture, and the user model is much harder to craft. In fact, one must study each of a whole set of applications, determining the peculiar needs and balances of each. Scientific computing emphasizes matrix algebra and partial differential equations; engineering computation emphasizes data reduction and formula evaluation; database query emphasizes optimum disk utilization. Each application must be understood.

Then they all have to be weighted:

- Across the entire application set
- Across the entire set of intended machine implementations
- Across the decades of lifetime that a new architecture must contemplate[5]

Likewise, as the design proceeds, the nascent result must be tested against the assumed characteristics of each user segment. When the design is complete and prototyped, the prototype must be tested by actual users from each segment.

So it is that I always assign my students in my advanced computer architecture course to do special-purpose architecture projects. They cannot offer gas and platitudes; the application and user analysis must be precise. And yet, they often do an excellent job of this, whereas they could not possibly devise an excellent unconstrained general-purpose architecture in the time available. The task is far easier than would be the in-depth analysis that a serious general-purpose design effort must make.

Software Design. The same paradox holds for software. Designing a special-purpose programming language is straightforward compared to the delicate balancing of expressive power, generality, and parsimony that one must seek in a general-purpose programming language. Restraint is so much easier to practice in the special-purpose design.

Spatial Design. The paradox holds as well for building spaces. Designing a superb bedroom is easier than designing superb public living spaces, precisely because the public spaces have so many more functions, so many more scenarios to be studied, and so many more furnishing options.[6]

Similarly, designing a specialized laboratory is easier than designing the public lobby of a computer science building.

Net

Since constraints are the designer's friend, if the task originally seems unconstrained, first think harder about what is really desired, about the user and use models, and you will probably find some narrowing constraints, to the benefit of both designer and user.

Notes and References

1. Wolff [2000], *Johann Sebastian Bach*, 387. When not sufficiently constrained by commission or available performance talent, Bach would sometimes adopt quite artificial constraints to stimulate his creativity. An example is the repeated use of the BACH motif (the sequence of notes B-flat, A, C, B-natural). I don't recommend the adoption by engineering or software of artificial constraints to stimulate creativity.
2. Revelations 22:16.
3. http://www.blueridgeparkway.org/linncove.htm, accessed on July 18, 2009.

4.

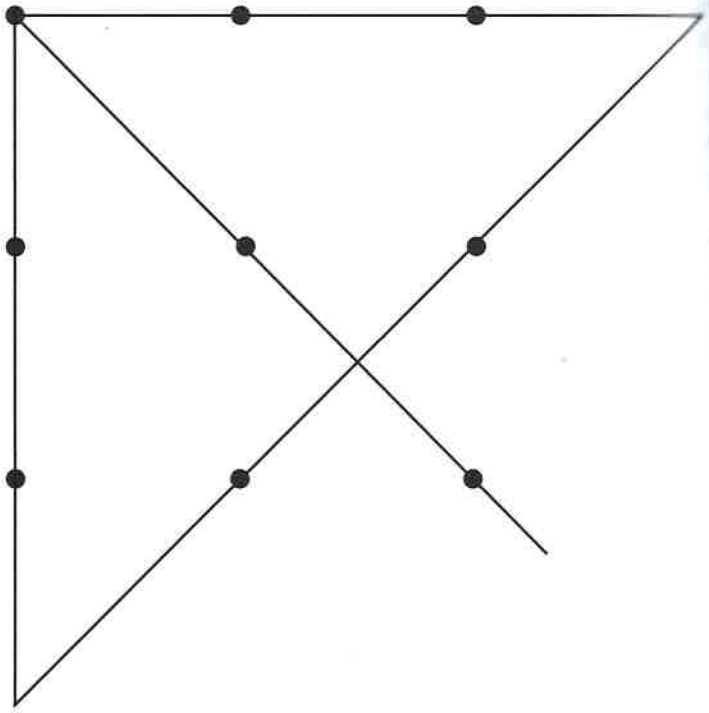

Figure 11-5 A solution to the nine-dots puzzle

5. We originally predicted that the System/360 architecture would have to live 25 years, through generations of implementations (Brooks [1965], "The future of computer architecture"). The forces for preserving programming languages, computer architectures, and operating system architectures are stronger than we then knew. IBM's z/90 still embodies System/360 architecture 45 years later, and the end is not yet in sight. Fortran (1956) use today is another good indicator of the strength of those persistence forces.
6. For the Kenwood House in Hampstead near London, Robert Adam (1728–1792) designed every detail of the furnishings, down to the doorknobs (*Kenwood Guidebook).*