

Memory

# Sixty years of Humanizing the Craft

## A conversation with Mel Conway

### HOW DO YOU MANAGE A TEAM OF DESIGNERS?

by MELVIN E. CONWAY

That kind of intellectual activity which creates a useful whole from its diverse parts may be called the *design of a system*. Whether the particular activity is the creation of software, the design of a mechanical system, or the organization of a business, the designer's role is to bring all the parts of the system together so that they function as a whole. The designer's task is to create a system which is more than the sum of its parts. The designer's task is to create a system which is more than the sum of its parts. The designer's task is to create a system which is more than the sum of its parts.

The design organization may or may not be involved in the construction of the system it designs. Frequently, in public affairs, there are policies which discourage a group's doing its own design. In such cases, the designer's role is to create a system which is more than the sum of its parts. The designer's task is to create a system which is more than the sum of its parts. The designer's task is to create a system which is more than the sum of its parts.

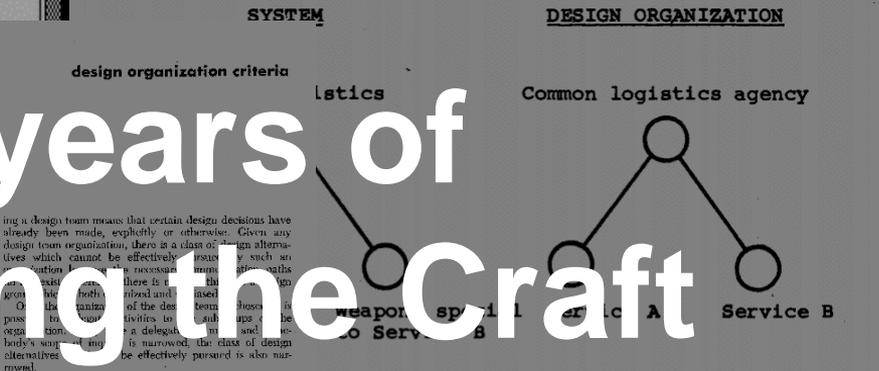
**stages of design**

The initial stages of design are concerned with the structure of the system. The designer's task is to create a system which is more than the sum of its parts. The designer's task is to create a system which is more than the sum of its parts. The designer's task is to create a system which is more than the sum of its parts.

1. Understanding of the problem and the system to be designed, placed by the sponsor and by the world's realities.
2. Achievement of a preliminary notion of the system's organization so that design task groups can be meaningfully assigned.

We shall see in detail later that the very set of organiza-

<sup>1</sup> A related, but much more comprehensive discussion of the behavior of system-designing organizations is found in John Kenneth Galbraith's, *The New Industrial State* (Boston, Houghton Mifflin, 1967). See especially Chapter VI, "The Technostructure."  
<sup>2</sup> For a discussion of the problems which may arise when the design activity takes the form of a project in a functional environment, see C. J. Middleton, "How to Set Up a Project Organization," *Harvard Business Review*, March/April, 1967, p. 73.



ing a design team means that certain design decisions have already been made, explicitly or otherwise. Given any design team organization, there is a class of design alternatives which cannot be effectively pursued by such an organization. In a necessary sense, the designer's task is to create a system which is more than the sum of its parts. The designer's task is to create a system which is more than the sum of its parts. The designer's task is to create a system which is more than the sum of its parts.

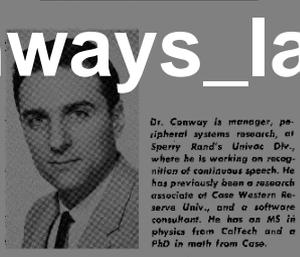
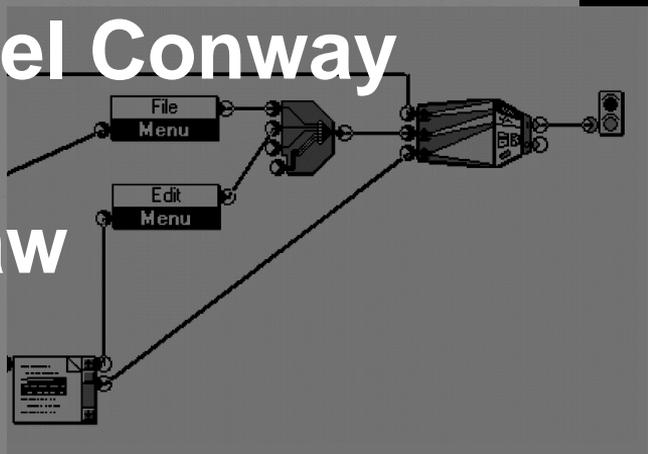
Once scopes of activity are defined, a coordination problem is created. Coordination among task groups, although it appears to lower the productivity of the individual in the small group, provides the only possibility that the separate task groups will be able to consolidate their efforts into a unified system design.

Thus the life cycle of a system design effort proceeds through the following general stages:

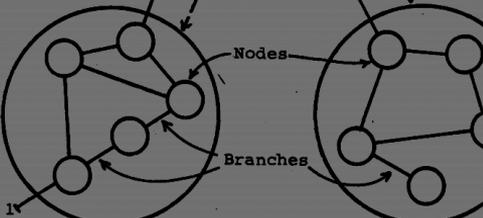
1. Drawing of team organization according to the design rules.
2. Choice of preliminary design concept.
3. Organization of the design task groups and delegation of tasks.
4. Coordination among delegated tasks.
5. Consolidation of subdesigns into a single design.

It is possible that a given design activity will not proceed straight through this list. It might conceivably regress upon discovery of a new, and obviously superior, design concept, but such an appearance of uncertainty is unflattering and the very act of voluntarily abandoning a creation is painful and expensive. Of course, from the

3a. A Weapon System



Dr. Conway is manager, peripheral systems research, at Sperry Rand's Univac Div., where he is working on recognition of continuous speech. He has previously been a research associate of Case Western Reserve Univ., and a software consultant. He has an MS in physics from Caltech and a PhD in math from Case.



# @conways\_law

**Incentives Affect the Product**

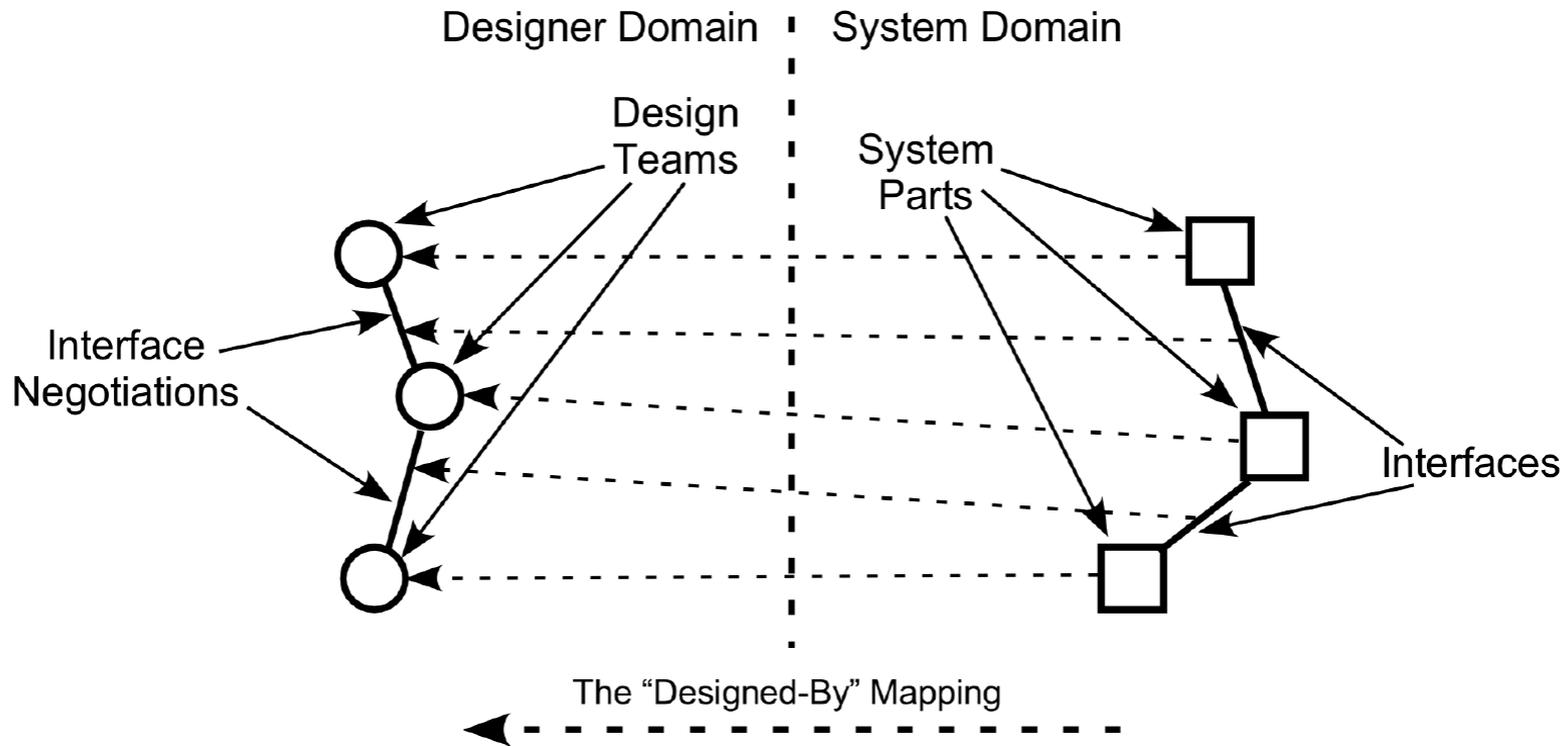
# Incentives Affect the Product

Lesson 1:

*You can make it even simpler if you keep working at it.*

Lesson 2:

*If you want the cleanest possible product find the simplest possible design before organizing to build, or else be prepared to reorganize.*



# Partition the Solution



# Partition the Solution

Lesson 3:

*Expressive domain-specific intermediate languages give the combined solution a lot more bang for the buck.*

**Static is Good**



# Static is Good

Lesson 4:

*Making application development accessible to a large number of people requires elimination of algorithms.*

Lesson 5:

*An effective application language presents a static parameterization of the implicit run-time algorithm.*

Lesson 6:

*One purpose of an application-development language is not to express algorithms, but to **hide** them.*

**Simplify the Developer's Life**





File Edit Search Run Windows

Apple // Instant Pascal®

©1985,1986 **THINK** Technologies, Inc.

Release 1.5

Engineered by:

Robert Swerdlow, Robert Herold

Robert Alpert, Russell Finn

Melvin Conway, Michael Byrne

Peter Maruhnic, Stephen Stein

Current program file:

(none)

Serial #6178635590

OK



Memory  
in use



# Simplify the Developer's Life

Lesson 7:

*Give the developer immediate feedback.*

Lesson 8:

*Don't make the developer distinguish between "programming" language and "execution" language*

**Humanize the Craft**

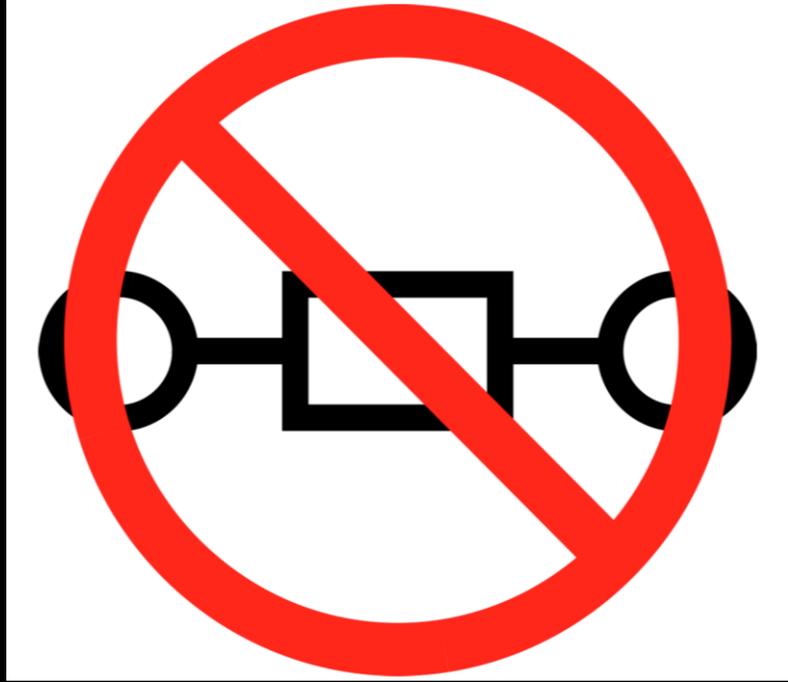
# Humanize the Craft

Lesson 9:

*Event-driven applications can be described with unidirectional flow diagrams.*

Lesson 10:

*The way to make application development universally accessible is to harness the hand-eye-brain system.*



# Humanize the Craft

Lesson 11:

*The input-process-output application-building model must be replaced by a transform-in-place model.*

Lesson 12:

*To simplify application development, tools must act like hands on tools.*

# **Six Hands-On Principles**

<p><b>Unity</b></p> <p>No translation Always on</p>	<p><b>Transparency</b></p> <p>Illusion: the tool is invisible. Your hands are on the working material</p>	<p><b>Continuity</b></p> <p>No surprises. Small changes produce predictable effects</p>
<p><b>Immediacy</b></p> <p>The brain immediately understands the result of each change</p>	<p><b>Interactivity</b></p> <p>The feedback you receive suggests your next action</p>	<p><b>Reversibility</b></p> <p>Undo</p>

**@conways\_law “Software as Child’s Play”**

# Epilogue

# Epilogue

- Static is Good
- Hands-On
- Transform In Place
- Six Design Principles for Hands-On Tools

# Epilogue

The Challenge:

*The developer must be able to build interactively any application whose components can be anywhere on the network and that is represented in its entirety on the user interface of a tool that conforms to all the hands-on design principles.*

