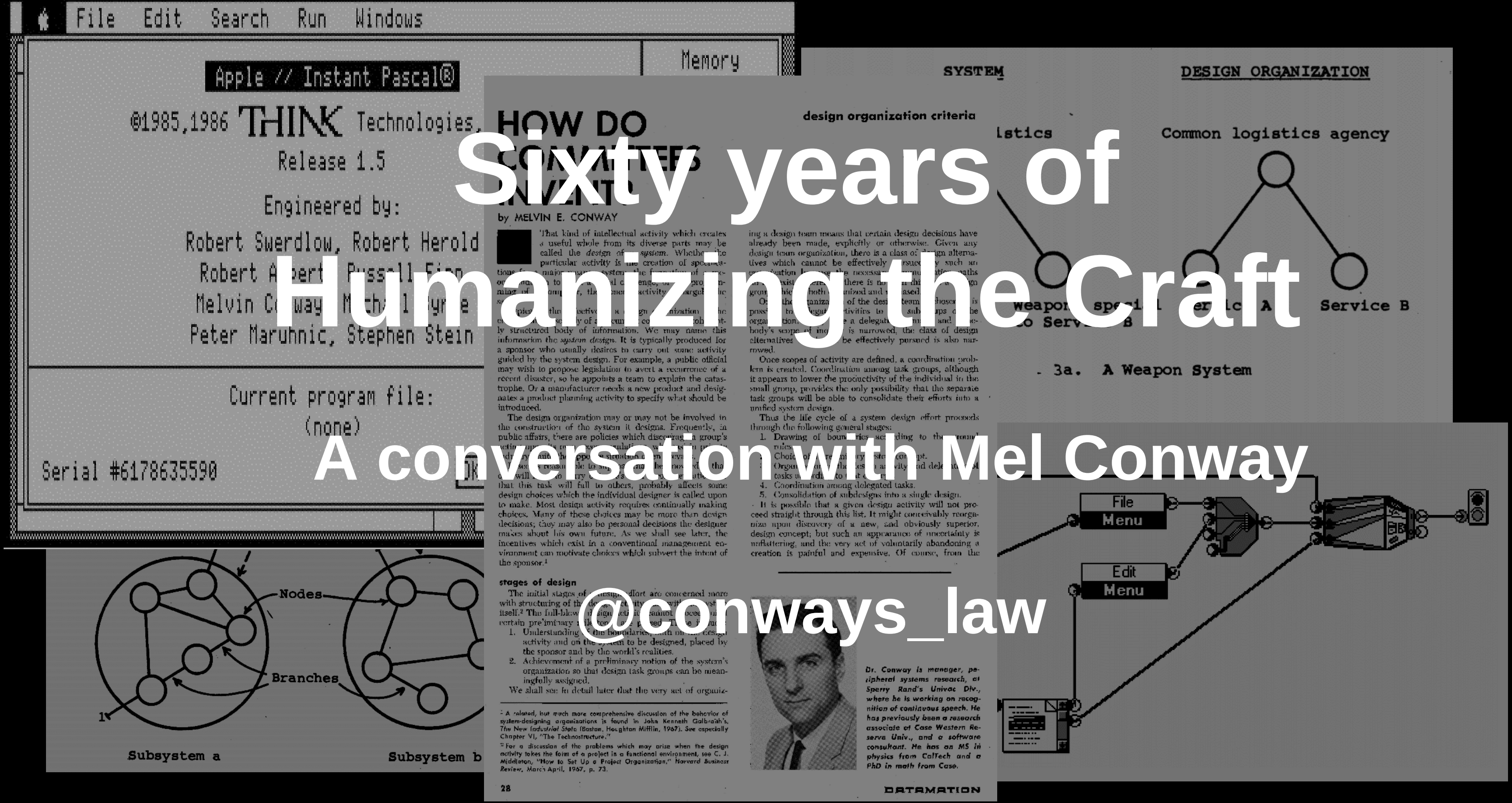
Sixty years of Humanizing the Craft

A conversation with Mel Conway

@conways_law



Incentives Affect the Product

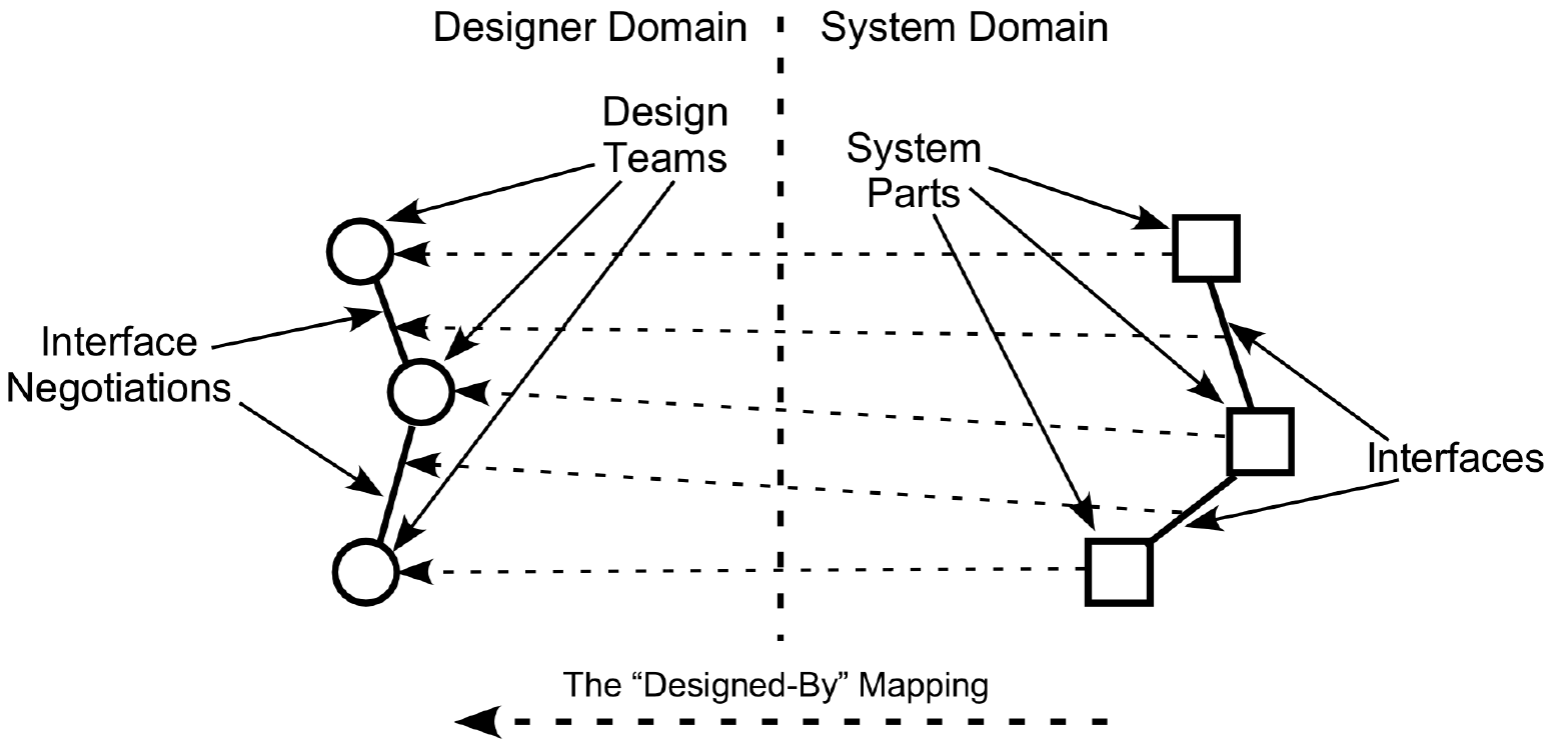
Incentives Affect the Product

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.


Designer Domain
System Domain
Design Teams
System Parts
Interface Negotiations
Interfaces
The “Designed-By” Mapping

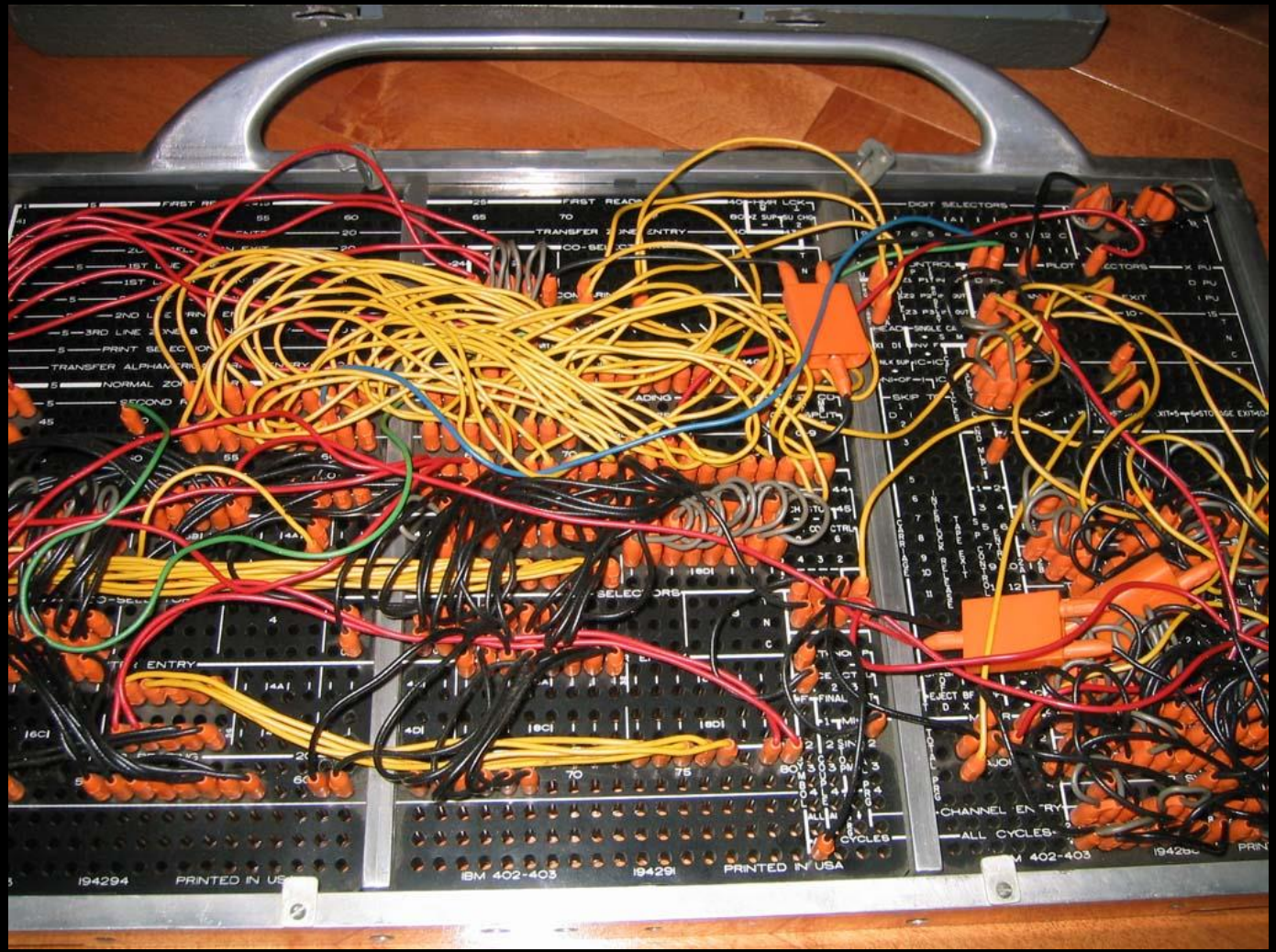
Partition the Solution

Partition the Solution

Lesson 3:

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

Static is Good


IBM 402-403
PRINTED IN USA
IBM 402-403
PRINTED IN USA

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


RESET
Rockwell


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
1
.5
0

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

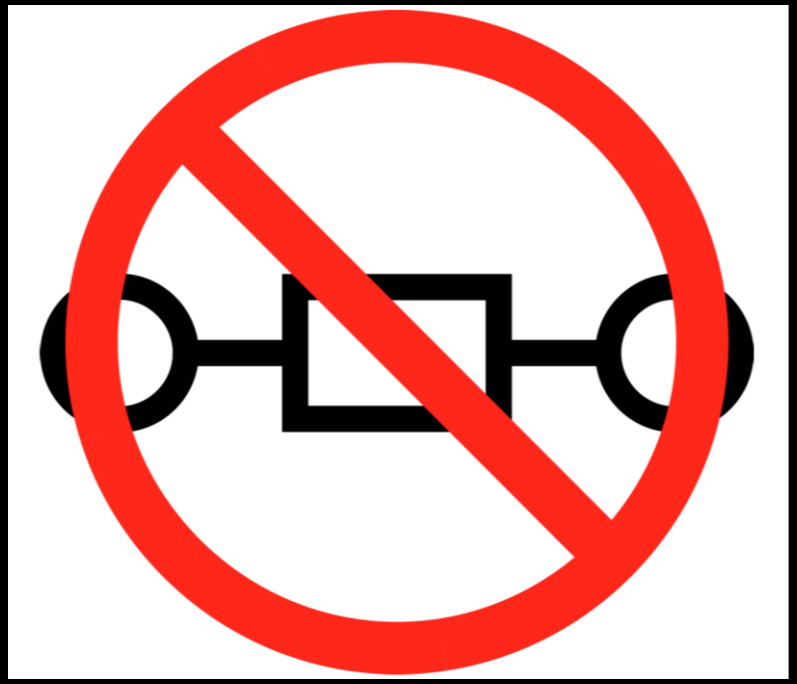
Humanize the Craft

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

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

@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

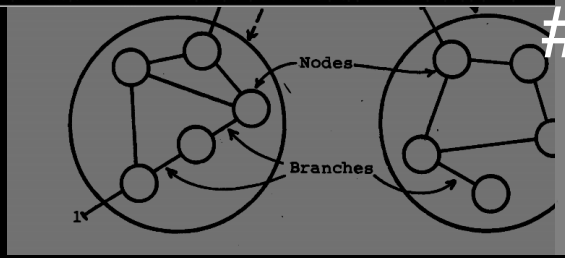
The Challenge:

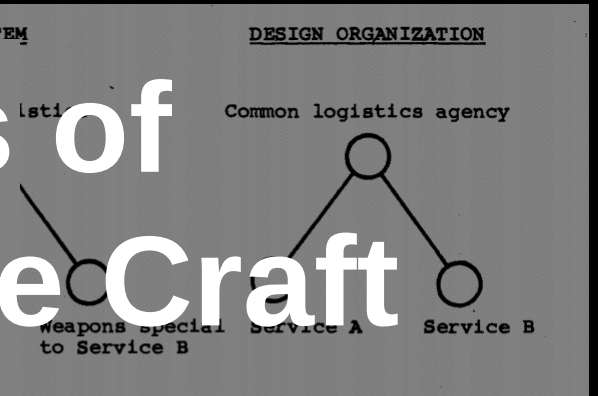
The developer must be able to build interactively any application whose components can be anywhere on the network and that is represented in its entirety on the user interface of a tool that conforms to all the hands-on design principles.

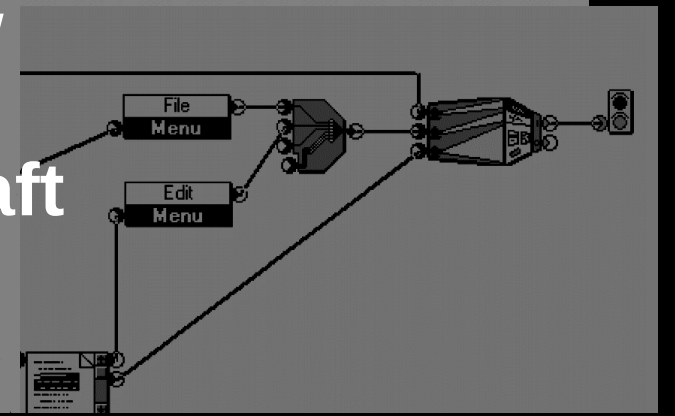
Sixty years of Humanizing the Craft

@conways_law

#HumanizeTheCraft







“Toward Simplifying Application Development, in a Dozen Lessons”