The Humane Dozen

Mel Conway

The twelve attributes below summarize my current thinking about the simplest possible way to think about, and build, interactive applications. This list arises from 60 years of thinking about simplicity. The major insights came when I stopped thinking like a computer scientist and started thinking like an anthropologist.

Simplicity seems to be the flip side of universal understanding.[1] That's because the key to simplicity is the key to universality: what Nature has invested heavily in building into every human: the hand-eye-brain system. In other words, the key to simplicity, as I have come to understand it, is to think about building software as a manual activity.

I haven't been able to separate thinking about the conceptual model of an application from thinking about the tools required to build that application. Another thing I haven't been able to separate is the set of twelve attributes itself. They can't be thought of in isolation. They are a set of interrelated guidelines for designing a language-tool synthesis. Once that synthesis exists it can be played with and evaluated. I have built one such example using a dataflow wiring metaphor[2] and am exploring its use now. Some of the items below might not even make sense by themselves until they are seen in the context of using such a holistically-designed tool.

[1] My models for this assertion are these three technologies: calendar, writing, and arithmetic. All three evolved from the obscure and complex domain of a priest class to being taught in elementary school.

[2] A 6 ½-minute silent annotated video illustrating these principles is at https://vimeo.com/275108662

Twitter: @conways_law

June 14, 2018
conway.mel@gmail.com

The Seven Attributes of a Hands-on Workflow

Hands on the working material

Consider this thought experiment. Imagine that you are a potter in the bowl-making business, but the individual potter's wheel technology does not exist. Instead what you have to do to make a bowl is write a bowl-turning script in a text editor, email it to China, and then wait for the bowl to be shipped back.

That is pretty much how we built software when I got started in the 1950s. Things are a little better now, but we're still a lot closer to the text-editor end of the hands-on spectrum than to the potter's wheel end. My goal, simply stated, is to move along that spectrum toward the potter's wheel.

The Hands-on Spectrum

The following seven attributes of a tool-language synthesis are what I consider necessary to move toward the hands-on end of the spectrum.

1. Immediate

Every modification made to the working material is immediately seen in its behavior.

2. Continuous

Small changes lead to predictable outcomes.

3. Interactive

The result of each change suggests the next change, like a child playing with a construction toy.

4. Transparent

The tool seems invisible and the artisan's hands are directly on the working material.

5. Inspectable

All parts of the application can be inspected at any time.

6. Modifiable

The artisan can change his or her work in midstream.

7. Reversible

A good UNDO means no regrets and no dead-ends.

Twitter: @conways_law

June 14, 2018
conway.mel@gmail.com

The Five Attributes of a Single-mode Workflow

Minimum mental gear-shifting

A *Mode* is a cognitive context you have to put yourself in to do a particular task, for example, being in the mind-set of command-line syntax in order to compose a command, or programming-language syntax in order to write code.

Switching between modes is taxing and error-prone. The more modes, the greater the cognitive load. The following five attributes minimize cognitive load, minimize switching cost, and enhance fluidity in the workflow.

Notice *Self-revealing* in particular. This is the antithesis of coding. Instead of being required to enter a syntactically correct expression, you choose from alternatives each of which has been explained. When this attribute is put into practice consistently and without exception it transforms the construction process.

1. Unified	"Source" and "Object" language representations are the same.
2. Self-revealing	Interfaces *present* and *explain* choices; they don't require formal constructions.
3. Symmetrical	The tool and the application being built are *side-by-side peers*, running at the same time. Your next action can be on either.
4. Always on	There is no concept of starting or stopping applications or components during construction. When you drag a component onto the workspace it is running.
5. Alive with your data	You can see the flow of data through the application moment by moment.

Twitter: @conways_law

June 14, 2018
conway.mel@gmail.com