Software Design and Architectures
SE-2 / SE426 / CS446 / ECE426

Lecture 1 : What Is It

This lecture introduces design and software design.

Design: the Impossible Task
Software may be a new thing in human history, but *design* is not. What is design? Etymology: *designo*, mark out: as one might mark out the foundation of a building on the ground, or pick out the lines of an image onto a wall.

As the term is used today, it's hard to pin down. But in designing we are reasoning about somthing we will build or adapt, before we build or adapt it, as if we had built or adapted it. Design is <u>using current information to predict a future state that will not come about unless their predictions are correct</u> [J C Jones, 1970] Note the inherent contradiction in this view. Design is "impossible", and tends to make people use suggestive metaphors and diagrams to "explain" what is happening.

Some sentences:
I never have to look at the manual: it's so well designed it seems to know what I want.
It's so hard to clean this thing: what were the designers thinking of?
That toy broke because of a design flaw: but this one was just cheaply made.
I don't think this is going to work: let's look at that part of the design again.
Ford vans share their basic design with Ford trucks; but Chrysler vans are more like cars.

Good and bad design is judged by properties ("ilities" - more later)
usability
maintainability
inherent robustness
reusability

Models and Views
Can we distinguish the "design" of a tool from the tool itself? Consider the problem of rusting rear door of my van. Future versions of this car might solve the problem by
a) using a different material or coating
b) changing the angle or shape of the entry to avoid catching snow and salt

So *yes*, because for example we can imagine *changing the design* of the car without really changing the car itself. But *no*, because we can't use, consider, think about the car without experiencing (the consequences of) its design.

However, we *can* consider the design independently from the construction process. Mostly this is done so that we can concentrate on certain features by ignoring others – including *not deciding about them*. (ref. impossible)

Only things which are built intentionally can be said to have a design, or be designed. When we speak carefully of natural phenomena we don't speak of *design* but of *properties* and *processes*.

Example: garden shed [Budgen]
Lumber company will branch out into prefab. sheds. Need a set of designs of various sheds, diff. sizes, diff. costs. Must minimize setup during mfg, to keep cost down. So constrain the design as
** use prefab. panels*
** use modular heights (for fixed-with boards)*

** reuse panels in multiple designs*
** use standard windows and doors*

The results might be communicated as:
- A provisional construction plan for each shed, directed at users
- Miniatures of all the panels and a miniature of each shred (proof of concept) *What has been ignored?*
- A full-size mock-up of one panel showing fasteners and (say) coatings. *What has been ignored?*

Note interaction of incomplete models: *multiple views*. This will be characteristic.

Note the separation between **what** and **how**. This will be characteristic.

Now *design* has two complementary meanings:
a) a (set of) models of a projected artefact, for reference during construction (and design!)
b) a plan for the construction, appearance, use of a projected artefact.

Among our artefacts are *tools*: and software is mostly a *tool* so we must think of building tools. Designing tools is even more self-refential a task than other design, because we are using tools to design and build tools.

Characteristically, when describing a designed thing we tend to speak in *future tense* as in "the system will respond by..." or maybe in the conditional as in "what would happen if we tried this design"? But of course even this is not quite true. No good verbal mood for this.

How can we *predict*?
intuitive experience, shared or individual
repeatable prediction method: measure, compare to standard results
methodical process: do this...

Patterns
Essential is the notion of sharing and communicating about design.
Recently we've begun to get excited about *patterns*, the

Testing

Process
So *design* is impossible because it requires us to describe and predict properties which will be correct only if they are correct. Experience, our own and others', replaces this impossible task.

A process may be descriptive (here's how it works) or prescriptive (here's what you should do). It may be hard to tell for a given account which it's intended to be; it may be that the author himself didn't know. But you should know what you think and what you read.

Extremely basic process: pure waterfall (note metaphor: inevitable descent) Often erected as a straw man. Best understood as a descriptive schematic. Feedback is always assumed.

Refinement: spiral [Boehm]

Notions: deliverable, stage ("phase, workflow, discipline")

What is Software?
Representing Design Decisions

Some Process and Design Myths [Pressman p17]

"We have a book full of standards, isn't that enough"
No, used? Complete? Modern?

"Can't we just add more staff to keep on track and improve?"
Famously no, Brooks: makes late later. Software equation: E varies as LOC^3 / t^4. Halve the development time, 16-ple the effort.

"General statement of objectives is enough to start coding"
No - see SE-1, requirements deficiency is key cause of cost overrun.

"Change can easily be accomodated because softrware is flexible"
Change happens. But flexibility must be designed in, it doesn't happen for free.

"Once we get the program to work, we're done".
The sooner you start writing, the longer it'll take. 50-70% effort after first delivery

"Until I get the program running I can't assess it's quality"
This is what design and process are all about – predicting qualities before construction

"The only deliverable for a successful project is the working program"
A successful project includes installed software, user documentation, and data.