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

Lecture 2 : Process and Model

This lecture considers *design* as both model and design process, and places them in the context of software development process generally.

Recall that design means simultaneously an idea, a model, and a process.

We should drop the notion that *design* is just ideas. We can't do *anything* rational, including building things, without thinking, without ideas. Our study is not psychological, nor metaphysical. To discuss designs and how to design for construction, we need concrete *evidence* of the ideas which precede construction. An artefact built without evidence of prior experiment, modeling, conscious process, we will just say "has no design".

And we'll define design thus:

> A *design* is a model or set of models of a projected construction made by a deliberate process.
> A *design process* is the deliberate process by which a design is produced.

This excludes: purely ideal "design", purely intuitive "design", "design" by iteration. More later.

Though we say *projected* of course the design doesn't cease to be a design after construction is begun or completed. Though the design may then be considered an historical artefact, or the basis of further (projected) construction, or the basis of documentation for the thing to be built.

What is a Model
Let's look harder at what *models* are.

We design and build *together* in a team. We assess designs (i.e. models) by evaluating their properties as models and inferring corresponding properties of the projected construction. Thus models must be *concrete* enough to be evaluated objectively, to be *related* to the projected construction, and to be *communicated* between designers. Typically it is *small*, *cheaper* and *easier to work with* than the 'real thing' – so that we can make *many models*.

A model might be a description or mathematical construction; or it might be a (physical) construction or a construction mimicking the planned artefact in some way.

Some examples of models.

Model Example 1
[Budgen p10] *Moving to a new house .. [is] a good illustration of what designing can involve. ... Measure the dimensions of various rooms in the house, obtain some suqared paper and use this for drawing up a scale plan showing all the rooms, doors, windows and so on. Meaure the dimensions of various items of futniteure and cut out cardboard shapes to represent these on the chosen scale. Together these form a represntaiton of the new house and its future contents that cna be considered as forming the* ***initial model****. ...*[p11] *The representation procided by the model ... is somewhat incomplete, since it is only two-dimensional.*

Why is this *concrete* enough to be evaluated? Because measurements can be made. Because we have the notion of *failure* and *success* in the model (i.e. placement of furniture). *Constraints* can be evaluated: perhaps "no stoves in the living room" or "piano in the library" are required. The *requirements* can be understood as a set of constraints which must be assessed against the model(s).

Why is this *related* to the projected construction? Because it's a scale drawing, so we know that measurements on the model correspond to measurements in the real world. Because a failure in the planned arrangement corresponds to a failure in the model. (Sometimes: see below.)

Why is this *communicable*? Because we can sit down and discuss it together, or show it to the movers. Because we can keep it in a drawer and consult it again.

What is *missing/ignored* in this model? The difficulty of acutally moving the furniture (the implementation plan). The question of whether the floor is strong enough to support the grand piano.

Might add *additional views* which would be new models, perhaps in a different medium.

Might improve the model by adding location of outlets and windows.

Might make a new model of a single room in three dimensions.

Might get some paint chips for comparison with furniture.

Model Example 2

In software...

Formal and Informal Models
A model may be more or less *formal*. This distinction relates to the degree to which *validating* and *evaluating* the model is predefined, algorithmic, mechanical.

This applies to processes too: in fact to all rational activity.

Some formal modeling styles in software: finite state machines; SDL; interface declarations; Rational UML; ERd's, Dataflow diagrams

Somewhat less formal modeling styles in software: UML 'sketches', architectural style diagrams

Rather informal: CRC cards, "design documents".

Design Process
Here the emphasis [Budgen, Pressman] is on the approximate processes of design *qua* design.

Modeling requires a process so we can see where we are, discuss "what to do next". We saw a "meaningless diagram" about it last time. Here is a better one.

[Budgen diagram]

A process has
- work products or *deliverables* (req. spec; functional spec.; "model"; "mismatch"; "blueprint")
- states
- transitions

The descriptions of the deliverables may be more or less detailed and/or formal. In general it may be possible to follow a process "in parallel" that is with several members of the team carrying out different steps, or being in different states, at a given time. This is a *management* or *planning* problem, more later.

The deliverables themselves may also be more or less details and/or formal. Note that this is not the same as formal *descriptions*.

Process doesn't refer to the actual steps really taken, but the structure and transition, the *account* of the process. We are not talking about history here but intent.

Prescriptive and Descriptive Processes
The Budgen diagram is not a complete formal process, nor is it prescriptive.

A *descriptive* process merely explains what happens empirically. *I watched them design and they seemed to be doing **this**.* A *descriptive* process regulates what happens when adopted. *You will design according to this process.* [Boehm]: "what should we do next" ; "how long should we do it for".

A descriptive process can nevertheless be followed or adopted; better than nothing, and perhaps can help to be rational about what we're doing. But recognize that it's based on empirical study, not necessarily tested.

An actual history can be explained or accounted for using a prescriptive process: interesting to see how well the process was actually followed, or to what extent a new prescription resembles some actual practice.

Software Design Process
[Somerville p 211]

This diagram shows the possible work products.

Development Process
Just as design *qua* design is produced by a process, so it fits into the larger activity called *development*.

Development processes appear similar to design processes, but with more products and more steps: they're bigger.

The term "process model" is sometimes heard, it means approximately what we're calling "process".

The *waterfall* process is the grand-daddy of process (models). It is really *descriptive*. It is called a *linear* process because apparently it (pre)(de)scribes a process in which each state accepts 1 input and produces 1 output, after which a unique state follows. but note feedback, and note that all could be proceeding in parallel. But then it wouldn't be *waterfall*: the metaphor is inevitable falling.

The *spiral* process is an attempt [due to Boehm] to discuss software process including feedback and progress. It is still a metaphor. There are many products.

The *open*

Exercise 1 due Monday Sep 15: describe a development and process model for doing a typical assignment in an undergraduate programming course (e.g. CS246). Enumerate the work products