CSC 484 Lecture Notes Week 3, Part 2
Details of the ID Process

I. **Relevant reading.**

A. Textbook Chapters 10 and 11.

B. Continuing with the papers of the fortnight:

1. *"Storyboarding: An Empirical Determination of Best Practices and Effective Guidelines"* by Truong, Hayes, and Abowd, from the University of Toronto and Georgia Tech; Proceedings of the 6th conference on Designing Interactive systems, 2006, ACM.
2. *"Developing Use Cases and Scenarios in the Requirements Process"*, by Maiden and Robertson, from the City University of London and the Atlantic Systems Guild; Proceedings of the 27th international conference on Software engineering (ICSE), 2005, ACM.

II. **Overview of Book Chapters 10, 11**.

A. These two chapters of the ID book cover territory that should be quite familiar to the software engineer:

1. Requirements analysis.
2. User-level design.
3. Prototyping.

B. The goals ID are fully in line with those of SE:

1. understand what users need;
2. construct a prototype to engage users;
3. evolve the prototype into design & implementation;
4. iterate as necessary.

C. Figure 1 is a summary of the process steps covered in these chapters, reorganized to reflect how we'll mostly do business in 484 projects:

III. **Introduction to Requirements Analysis (Section 10.1).**

A. Definition of user requirements.

B. Importance of gathering requirements.

C. Techniques to gather requirements.

D. Different requirements representations.

IV. **What, How, and Why? (Section 10.2).**

A. *Precisely* the same goals as in SE:

1. Capture requirements sufficiently well to start design.
2. Don't let fluctuating requirements slow down the process.

B. Goals achieved in two ways:

1. Focus on requirements first, postponing time-consuming design ***(traditional)***.
2. Focus on small requirements pieces, each with rapidly doable design ***(agile)***.

C. The ultimate in agility ...

1. Be agile enough to know when to go traditional.
2. When integrated design/implementation impedes progress, focus on requirements alone.

Establish[1] Requirements
- Storyboards
- Scenarios
- User-Level Model *(optional)*
  - *Use Cases*
  - *Volere Shell*
  - *Objects/Operations*

Design
- Conceptual Design
- Physical UI Design
- Concrete Product Design *(optional)*
  - *UML diagrams*
  - *Software APIs*
  - *Design of hardware, or other non-software components*

Construct Prototype
- Software
- Hardware, or other non-software media

Figure 1: The CSC 484 Development Process.

V. **What are requirements? (Section 10.3).**

A. More review of SE topics.

B. Bottom line definition of a requirement is a ***statement of fact***.

VI. **Different kinds of requirements (Section 10.3.1).**

A. ***Functional*** -- what the artifact does.

B. ***Non-Functional*** -- characteristics of the artifact, its development, its users.

C. The book's treatment is comparable to SE treatments, *plus usability*.

D. Specifically, enhance the SE process by adding *usability goals* and *user experience goals* to the coverage of non-functional requirements.

VII. **Data gathering methods (Section 10.4).**

A. *Interviews.*

B. *Focus groups.*

C. *Questionnaires.*

D. *Direct and indirect observation.*

E. *Studying domain-specific documentation.*

---

[1] Effective synonyms for "establish" in this context include: "gather", "capture", "elicit", "acquire", "identify", and "analyze". For choosing which of these terms to use, euphoneousness is really as good a criterion as any, given that subtle distinctions in word connotation are not all that enlightening (e.g., Section 10.2.4).

F. *Researching similar products.*

VIII. **Contextual Inquiry, Other Guidelines.**

A. Section 10.4.1 not particularly useful (IMO).

B. Section 10.4.2 repeats what's been said already.

IX. **Analysis, interpretation, presentation (Section 10.5).**

A. This is a very cursory treatment of SE topics.

B. Volere shell is yet another requirements notation, and not a particularly useful one at that (in my judgment).

X. **Task description, analysis (Sections 10.6, 10.7).**

A. Again, these are all well-known SE techniques.

B. Here are three important amendments to book's coverage:

1. Storyboards can come first, i.e., before scenarios.
2. Scenarios have *both* pictures and prose, not just prose alone.
3. Prototyping can commence *without* formal modeling, e.g., without UML use cases.

XI. **Introduction to design, prototyping, and construction. (Chapter 11).**

A. Again, this is very familiar territory in SE (EE, ***X***E, for ***X*** = aero, civil, mechanical, etc.).

B. The book's notion of *conceptual design* is pretty fluffy.

C. It presents some useful discussion of storyboarding and prototyping; the current class readings provide additional detail.

D. However Chapter 11 essentially punts on concrete design and construction, i.e., it leaves the subjects to other discipline-specific books and readings.

XII. **Prototyping and construction (Section 11.2).**

A. What is a prototype? (Section 11.2.1)

1. A reduced-functionality version of a product.
2. Allows user to interact with it.

B. Why Prototype? (Section 11.2.2)

1. Helps all stakeholders understand the product.
2. Helps achieve "full user engagement" during product design.

XIII. **Low-fidelity prototyping (Section 11.2.3).**

A. Doesn't look much like finished product.

B. Simple, cheap, and quick to produce.

C. Storyboards are considered by some to be a form of low-fidelity prototype.

1. However, storyboards do not typically afford the opportunity for significant end user interaction.
2. Such interaction is considered by many to be a fundamental property of a prototype.

D. Storyboarding activities include

1. *Sketching* -- embrace the wonders of clip art.
2. *Index cards* -- not my style, but use it if it works for your team.

3. *Wizards of Oz* -- humans sitting behind the scenes to simulate prototypical behavior.

XIV. **High-fidelity prototyping (Section 11.2.4).**

A. Looks much like the finished product.

B. Recall the balancing act described in last week's notes --
   1. Build a prototype as rapidly as possible.
   2. Include as much of what the user cares about as possible
   3. Leave out as much of the time-consuming implementation details as possible.

XV. **Compromises in prototyping (Section 11.2.5).**

A. *Horizontal prototyping* -- wide range of functions with little detail.

B. *Vertical prototyping* -- much detail for only a few functions.

XVI. **Construction: from design to implementation (Section 11.2.6).**

A. The authors squeeze into one half page the subject of several courses in Computer Science and Software Engineering, not to mention enumerable courses in other areas of science and engineering.

B. Construction is clearly not the focus of this text.

C. Also, I believe the discussion in the "Dilemma" box on page 539 is off the mark.
   1. The distinction between throw-away and evolutionary prototypes is important.
   2. However, the book fails to mention that truly evolutionary prototypes make sense for very few types of artifact other than software.

XVII. **Conceptual design (Section 11.3).**

A. By the book's admission, there is no definitive characterization of a conceptual model.

B. Neither is there a single specific artifact for the conceptual model.

C. Rather, elements of conceptual modeling are embodied in other concrete artifacts of the process, including:
   1. *The general user requirements*, where stakeholders wants and needs are described
   2. *Non-functional requirements*, which are formulated by understanding and empathising with stakeholders
   3. *Scenarios, story boards, and prototypes*, where operational concepts are embodied
   4. *Concrete design and implementation*, where concepts are concretely realized

D. In my opinion, much of the book's discussion about conceptual design and metaphor can be summed up in the following simple observations:
   1. Present ideas in ways that are understandable to the users.
   2. Explore alternative forms of interaction, suitable for users' tasks.

E. Other chapters in the book present specific guidelines to help achieve these ends, in particular chapters 3 and 6; we will cover these chapters in coming weeks.

XVIII. **Concrete design (Section 11.4).**

A. As with construction, the book punts on concrete design.

B. It notes that earlier chapters provide some specific guidance for the design process, in the areas of interface types and human cognitive factors.

C. Otherwise, the book defers to others to provide specific theories and techniques in the many disciplines involved in the design of interactive products.

XIX. **Scenarios in design (Section 11.5).**

A. These are well-established ideas in software engineering.

B. To reiterate a point made earlier in these notes, scenarios should most definitely include both pictures and prose.

XX. **Prototypes in design**

A. Again, these are well-established ideas in SE.

B. ***Generating storyboards from scenarios (Section 11.6.1).***

1. In my experience, it's mostly the other way around.
2. But do whatever works for your team.

C. ***Generating card-based prototypes (Section 11.6.2).***

1. I my view, these are utterly anachronistic.
2. But again, do what works for your project team.

D. ***Prototyping physical design (Section 11.6.3).***

1. This is what software engineers do all the time, particularly the agile ones.
2. I.e., they evolve the rough ideas into more concrete ideas.
3. E.g., evolve UI sketches, into UI screens, into working UI prototypes.

XXI. **Tool support (Section 11.7).**

A. There are plenty of code-level tools out there, in commercial and open-source IDEs.

B. There are also some more purely prototyping tools, such as Flash.

C. DENIM, and more recent research tools like it, have some interesting ideas.

D. However at present, there is still no practically usable tool that has well integrated functionality for story-boarding, prototyping, design, and implementation.