CSC 402, Lecture Notes Week 4

Review of Phase 1 Pre-Release

Planning for Friday's Phase 1 Release

Requirements Process Details

I. Weekly Lecture/Lab Overview

A. *Monday*:

1. Go over Milestone 4
2. Discuss evaluation comments on Phase 1 pre-release

B. ***Wednesday:***

0. New Task 0 for Milestone 4.
1. Discuss client correspondence for Phase 1.
2. Class-wide walk-through of requirements.
3. Stress test client forum.
4. Discuss using Trac wiki ticketing system.

C. ***Friday:***

1. Dress rehearsal of requirements release.
2. Fine tune requirements content and style.
3. Time permitting, discuss further requirements process topics.

II. Milestone 4 and Pre-Release Eval

A. For M4, see

```
~gfisher/classes/402/
   handouts/milestone4.html
```

B. For pre-release eval, see

```
release/alpha/scheduler/
   administration/evaluations/requirements/
      10oct11.html
```

III. Client Explanation for Phase 1 Release

A. What we’ve learned from interviews:

1. Strong need exists for proposed product.
2. Fair number of department similarities.
3. Some important differences.

Client Explanation, cont'd

4. Wide interest in:
   a. automatic schedule generation
   b. automating PeopleSoft data entry
   c. ***What else ... ?***

Client Explanation, cont’d

5. Common to many departments:
    a. Managing instructor info
    b. Managing course info
    c. Managing room info
    d. ***What else ... ?***

Client Explanation, cont'd

6. Variability among departments:
   a. How much to weight instructor prefs
   b. How much control over rooms
   c. ***What else ... ?***

IV. What MUST Happen by 4PM Today:

- VM storage issue resolved.
- ***All*** Visio source committed.
- Work breakdown fully completed.

V. Systematically Addressing Clients' Needs

A. Extract features from interviews.

B. Provide tracability from interview features to corresponding segments of the requirements.

VI. Informal Feature Extraction

A. Any brand new features?

B. Any preconceived features to modify?

C. Any preconceived features to remove?

Informal Feature Extraction, cont’d

D. We have done some of this already.

E. Next week we’ll get more systematic.

VII. **Systematic Feature Extraction**

A. Process goes like this:

1. Identify features in transcript text & docs.
2. Place each feature in functional hierarchy.
3. Determine UI for each feature.
4. Write use case scenarios.
5. Generate walk-thru slides.
6. Create tracability links.

Systematic Feature Extraction, cont’d

B. Involves thorough analysis of transcripts.

C. Provides concrete evidence that we’ve considered client input carefully.

VIII. **Identify features**

A. Requires careful human interpretation.

B. Separate functional, non-functional features.

C. Idea is to spot software features within over-all interview conversation.

D. Features often prefaced with "We need ...", "We’d like ...", or similar language.

IX. Place in Functional Hierarchy

A. As discussed in 308 notes, the command hierarchy is embodied in:

a. UI command/data structure.

b. Requirements organization.

c. Existing prototype model.

Place in Functional Hierarchy, cont’d

B. An identified new feature may:

   a. fit into existing functionality,

   b. require creation of new functionality,

   c. require reorganization of functionality.

X. Determine UI for identified feature

A. May covered by existing UI.

B. May require UI modification, upgrade.

C. May require new UI design.

XI. **Write Use Case Scenarios**

A. Core of the methodology.

B. We’ve reviewed many existing examples.

C. We’ve done started new scenarios in Phase 1.

XII. **Generate Walk-Thru Slides**

A. This has been major focus in Phase 1.

B. Hopefully we can partially automate scenario-to-slide generation process.

XIII. Create Tracability Links

A. Define link points in transcripts.

B. Define target points in requirements.

C. Use HTML refs as concrete implementation.

XIV. **Systematic Analysis Example**

A. Consider this statement from CSC client:

"We need to be able to mark certain blocks of time as unavailable for classes. For example, in the Computer Science department, we don’t want to schedule any classes on MWF 1-2PM, at least not for tenure-track faculty."

Systematic Analysis Example, cont'd

B. General strategy for this particular case:

1. Use existing instructor time pref UI to auto-fill read-only zero's in the time pref screen.

2. Add a new scheduler command that allows time blocks to be marked as unavailable.

Systematic Analysis Example, cont'd

C. Regarding the *"at least not for tenure-track faculty"* aspect, we can:

1. Add a data feature for instructors.
2. Values: "tenure-track", "lecturer", "student".
3. Provide time-block option to select who blocks apply to.
4. ***Run these ideas by the client*** for feedback.

XV. Process Details for Example Feature

A. Identified as "feature" by prose analysis, in particular the "We need ... " language.

Process Details, cont’d

B. Placement in hierarchy:

1. A new command is added somewhere in the scheduler’s command repertoire.
2. Existing data values are used for displaying in instructor time prefs.

Process Details, cont'd

C. UI design has two aspects:

1. Addition of new command in proper place.
2. Addition of some form of explanatory help for instructor to know why times are auto-blocked out.

Process Details, cont’d

D. Scenario details and slides to be worked out.

E. Concrete example of trace links:

Process Details, cont'd

```
<strong><em>
We need ... for tenure-track faculty.
<br>
Links to:
<a href= "../../instructor-time-prefs.html
                    #blocked-out-times">
    Instructor Time Prefs, Blocked Out Times
</a>
<br>
<a href= "../../scheduler-time-blocking.html>"
    Scheduler Blocking Out Selected Times
</a>
</em></strong>
```