CSC 307

Intro to the Course

I. Materials for weeks 1 and 2:

I. Materials for weeks 1 and 2:

A. Syllabus.

I. Materials for weeks 1 and 2:

A. Syllabus.

B. Project description.

I. Materials for weeks 1 and 2:

A. Syllabus.

B. Project description.

C. Writeup for Milestones 0 and 1

I. **Materials for weeks 1 and 2:**

A. Syllabus.

B. Project description.

C. Writeup for Milestones 0 and 1

D. Specification document outline.

I. **Materials for weeks 1 and 2:**

A. Syllabus.

B. Project description.

C. Writeup for Milestones 0 and 1

D. Specification document outline.

E. Milestone 1 example.

I. **Materials for weeks 1 and 2:**

A. Syllabus.

B. Project description.

C. Writeup for Milestones 0 and 1

D. Specification document outline.

E. Milestone 1 example.

F. SVN basics.

II. Scheduling first two weeks.

II. Scheduling first two weeks.

A. First day (Mon).

II. Scheduling first two weeks.

A. First day (Mon).

1. In Lecture:

II. Scheduling first two weeks.

A. First day (Mon).

1. In Lecture:

a. Tour of syllabus and other handouts.

II. Scheduling first two weeks.

A. First day (Mon).

   1. In Lecture:

      a. Tour of syllabus and other handouts.

      b. Intro to general SE concepts.

II. Scheduling first two weeks.

A. First day (Mon).

1. In Lecture:
    a. Tour of syllabus and other handouts.
    b. Intro to general SE concepts.

2. In Lab:

II. Scheduling first two weeks.

A. First day (Mon).

1. In Lecture:
    a. Tour of syllabus and other handouts.
    b. Intro to general SE concepts.

2. In Lab:
    a. Form project teams.

II. **Scheduling first two weeks.**

A. First day (Mon).

1. In Lecture:
    a. Tour of syllabus and other handouts.
    b. Intro to general SE concepts.

2. In Lab:
    a. Form project teams.
    b. Start working on projects.

Scheduling, cont'd

B. Second day (Wed):

Scheduling, cont’d

B. Second day (Wed):

   1. Lecture on intro to SE.

Scheduling, cont’d

B. Second day (Wed):

1. Lecture on intro to SE.
2. Lab on setting up project repo.

Scheduling, cont’d

C. Third day (Fri):

Scheduling, cont'd

C. Third day (Fri):

   1. Lecture on details of the project.

Scheduling, cont’d

C. Third day (Fri):

   1. Lecture on details of the project.
   2. Customer interviews in lab.

Scheduling, cont'd

D. Fourth day (Mon, Week 2):

Scheduling, cont'd

D. Fourth day (Mon, Week 2):

   1. Lecture on software requirements.

Scheduling, cont’d

D. Fourth day (Mon, Week 2):

   1. Lecture on software requirements.

   2. Second round of customer interviews in lab.

Scheduling, cont'd

E. Week 3 and beyond.

Scheduling, cont’d

E. Week 3 and beyond.

   1. Mostly normal lectures.

Scheduling, cont’d

E. Week 3 and beyond.

1. Mostly normal lectures.
2. Lab meetings as described in syllabus.

Syllabus

Syllabus

- **Instructor**

Syllabus

- **Instructor**

  Gene Fisher

  14-210, gfisher@calpoly.edu

Syllabus

- **Instructor**

  Gene Fisher

  14-210, gfisher@calpoly.edu

  Office Hrs:
  MWF 4-5, Tu 9-11, by appt

Syllabus, cont’d

- **Course Objectives**

Syllabus, cont'd

- **Course Objectives**
- **Class Materials**

Syllabus, cont’d

- Course Objectives
- Class Materials
- Activities

Syllabus, cont’d

- Course Objectives
- Class Materials
- Activities
- Evaluations

Syllabus, cont’d

- Course Objectives
- Class Materials
- Activities
- Evaluations
- Bi-Weekly Activity Reports

Syllabus, cont’d

- **How to Submit Project Work**

Syllabus, cont’d

- How to Submit Project Work
- Team Work

Syllabus, cont’d

- How to Submit Project Work
- Team Work
- Computer Work

Syllabus, cont’d

- How to Submit Project Work
- Team Work
- Computer Work
- Lecture, Lab, Milestone Scheduling

Class Project

Class Project

- This year’s project is a testing tool.

Class Project

- This year's project is a testing tool.
- For use in Computer Science dept.

Class Project

- This year’s project is a testing tool.
- For use in Computer Science dept.
- Major features:

Class Project

- This year’s project is a testing tool.
- For use in Computer Science dept.
- Major features:
    1. A question bank

Class Project

- This year’s project is a testing tool.
- For use in Computer Science dept.
- Major features:
  1. A question bank
  2. Semi-automated test generation

Class Project

- This year’s project is a testing tool.
- For use in Computer Science dept.
- Major features:
    1. A question bank
    2. Semi-automated test generation
    3. Electronic test taking

Class Project

- This year's project is a testing tool.
- For use in Computer Science dept.
- Major features:
  1. A question bank
  2. Semi-automated test generation
  3. Electronic test taking
  4. Semi-automated test grading

Class Project

- This year’s project is a testing tool.
- For use in Computer Science dept.
- Major features:
    1. A question bank
    2. Semi-automated test generation
    3. Electronic test taking
    4. Semi-automated test grading
- We’ll walk through paper handout ...

Question bank

Question bank

- different kinds of questions

Question bank

- different kinds of questions
- convenient UI

Question bank

- different kinds of questions
- convenient UI
- questions have attributes

Question bank

- different kinds of questions
- convenient UI
- questions have attributes
- all the standard types

Question bank

- different kinds of questions
- convenient UI
- questions have attributes
- all the standard types
- questions contain graphics

Question bank

- different kinds of questions
- convenient UI
- questions have attributes
- all the standard types
- questions contain graphics
- questions have program code

Semi-automated test generation

Semi-automated test generation

- simple generate

Semi-automated test generation

- simple generate
  - user enters a few parameters

Semi-automated test generation

- simple generate
  - user enters a few parameters
  - presses the "Generate" button

Semi-automated test generation

- simple generate
  - *o* user enters a few parameters
  - *o* presses the "Generate" button
- more advanced generation

Semi-automated test generation

- simple generate
  - user enters a few parameters
  - presses the "Generate" button
- more advanced generation
  - user enters question details

Semi-automated test generation

- simple generate
  - user enters a few parameters
  - presses the "Generate" button
- more advanced generation
  - user enters question details
  - some kind of table UI

Semi-automated test generation

- simple generate
  - user enters a few parameters
  - presses the "Generate" button
- more advanced generation
  - user enters question details
  - some kind of table UI
  - presses the "Generate" button

Semi-automated test generation

- simple generate
  - *o* user enters a few parameters
  - *o* presses the "Generate" button
- more advanced generation
  - *o* user enters question details
  - *o* some kind of table UI
  - *o* presses the "Generate" button
- user can edit after generation

Electronic test taking

Electronic test taking

- students take tests electronically

Electronic test taking

- students take tests electronically
- taker has convenience features

Electronic test taking

- students take tests electronically
- taker has convenience features
- taking modes -- proctored, take-home, practice

Electronic test taking

- students take tests electronically
- taker has convenience features
- taking modes -- proctored, take-home, practice
- there’s a proctoring UI

Semi-automated test grading

Semi-automated test grading

- all types of question can be auto graded, at least partially

Semi-automated test grading

- all types of question can be auto graded, at least partially
- user can edit graded test

Semi-automated test grading

- all types of question can be auto graded, at least partially
- user can edit graded test
- user can add written comments

Semi-automated test grading

- all types of question can be auto graded, at least partially
- user can edit graded test
- user can add written comments
- graded tests are available to students

Additional Observations about the Project

Additional Observations about the Project

- TestTool is not an AI program

Additional Observations about the Project

- TestTool is not an AI program
- Primary setting is the CSC department

Additional Observations about the Project

- TestTool is not an AI program
- Primary setting is the CSC department
- Initially neutral on desktop vs browser UI

Milestone 0

Milestone 0

- **Due:** Friday 1st week, 5PM

Milestone 0

- **Due:** Friday 1st week, 5PM
- **Tasks:**

Milestone 0

- **Due:** Friday 1st week, 5PM
- **Tasks:**
  1. Form team

Milestone 0

- **Due:** Friday 1st week, 5PM
- **Tasks:**
  1. Form team
  2. Determine team governance

Milestone 0

- **Due:** Friday 1st week, 5PM
- **Tasks:**

1. Form team
2. Determine team governance
3. Brainstorm about tool features

Milestone 0

- **Due:** Friday 1st week, 5PM
- **Tasks:**
  1. Form team
  2. Determine team governance
  3. Brainstorm about tool features
  4. Start search for related tools

Milestone 0

- **Due:** Friday 1st week, 5PM
- **Tasks:**
  1. Form team
  2. Determine team governance
  3. Brainstorm about tool features
  4. Start search for related tools
  5. Start customer question list

Milestone 0 Deliverables

Milestone 0 Deliverables

- These project files:

Milestone 0 Deliverables

- These project files:
  - `governance.html`

Milestone 0 Deliverables

- These project files:
  - `governance.html`
  - `work-breakdown.html` *(1st of many drafts)*

Milestone 0 Deliverables

- These project files:
  - `governance.html`
  - `work-breakdown.html` *(1st of many drafts)*
  - `customer-questions.html` *(1st draft)*

Milestone 0 Deliverables

- These project files:
  - `governance.html`
  - `work-breakdown.html` *(1st of many drafts)*
  - `customer-questions.html` *(1st draft)*
- Files are checked-in and released.

Milestone 0 Deliverables

- These project files:
  - `governance.html`
  - `work-breakdown.html` *(1st of many drafts)*
  - `customer-questions.html` *(1st draft)*
- Files are checked-in and released.
- Use templates in 307 `handouts` directory.

Milestone 0 Preparation in Today’s Lab

Milestone 0 Preparation in Today's Lab

- Team librarian sets up project repo.

Milestone 0 Preparation in Today's Lab

- Team librarian sets up project repo.
- All team members check it out.

Milestone 0 Preparation in Today’s Lab

- Team librarian sets up project repo.
- All team members check it out.
- This coming Friday we’ll go over deliverable submission details.

Specification Document Outline

Specification Document Outline

1. Introduction

Specification Document Outline

1. Introduction

2. Functional Requirements

Specification Document Outline

1. Introduction

2. Functional Requirements

3. Non-Functional Requirements

Specification Document Outline

1. Introduction
2. Functional Requirements
3. Non-Functional Requirements
4. Developer Overview

Specification Document Outline

1. Introduction

2. Functional Requirements

3. Non-Functional Requirements

4. Developer Overview

5. Formal Specifications

Specification Document Outline

1. Introduction
2. Functional Requirements
3. Non-Functional Requirements
4. Developer Overview
5. Formal Specifications
6. Rationale

Specification Document Outline

1. Introduction

2. Functional Requirements

3. Non-Functional Requirements

4. Developer Overview

5. Formal Specifications

6. Rationale

A., B. Possible Appendices ...

III. What is software engineering?

III. What is software engineering?

A. The *disciplined* creation of software.

III. What is software engineering?

A. The *disciplined* creation of software.

B. Principles of scientific problem solving applied.

III. **What is software engineering?**

A. The *disciplined* creation of software.

B. Principles of scientific problem solving applied.

1. Define problem before solution.

III. **What is software engineering?**

A. The *disciplined* creation of software.

B. Principles of scientific problem solving applied.

1. Define problem before solution.

2. "Divide and conquer".

What is SE, cont’d

C. Principles of engineering are applied.

What is SE, cont’d

C. Principles of engineering are applied.

  1. Using formal mathematics.

What is SE, cont’d

C. Principles of engineering are applied.

1. Using formal mathematics.
2. Formally verifying solution.

IV. The different types of software.

IV. The different types of software.

A. Three broad categories:

IV. The different types of software.

A. Three broad categories:

1. *End-user software*

IV. The different types of software.

A. Three broad categories:

1. *End-user software*

2. *System software*

IV. **The different types of software.**

A. Three broad categories:

1. *End-user software*

2. *System software*

3. *Embedded software*

Types of software, cont’d

B. Two other categories based on clientele:

Types of software, cont’d

B. Two other categories based on clientele.

1. *Off-the-shelf*, or *open market*

Types of software, cont'd

B. Two other categories based on clientele.

1. *Off-the-shelf*, or *open market*

2. *Custom*, or *bespoke*

Types of software, cont'd

B. Two other categories based on clientele.

1. *Off-the-shelf*, or *open market*

2. *Custom*, or *bespoke*

C. In 307, we build *custom end-user* software.

V. The people involved with software.

A. The following are software "stakeholders":

1. ***end users***
2. ***customers***
3. ***domain experts***
4. ***analysts***

Software people, cont'd

5. ***implementors***
6. ***testers***
7. ***managers***
8. ***visionaries***
9. ***maintainers and operators***
10. ***other interested parties***

Software people, cont’d

B. First four groups work together.

C. Frequently, implementation team does not participate in the requirements spec.

Software people, cont'd

D. In 1st half of 307, you're primarily analysts, secondarily domain experts and end users.

Software people, cont'd

D. In 1st half of 307, you're primarily analysts, secondarily domain experts and end users.

E. In 2nd half, you're software designers and implementors.

VI. The software development process.

VI. The software development process.

A. Proper engineering uses an orderly process.

VI. The software development process.

A. Proper engineering uses an orderly process.

B. Figure 1 depicts major steps.

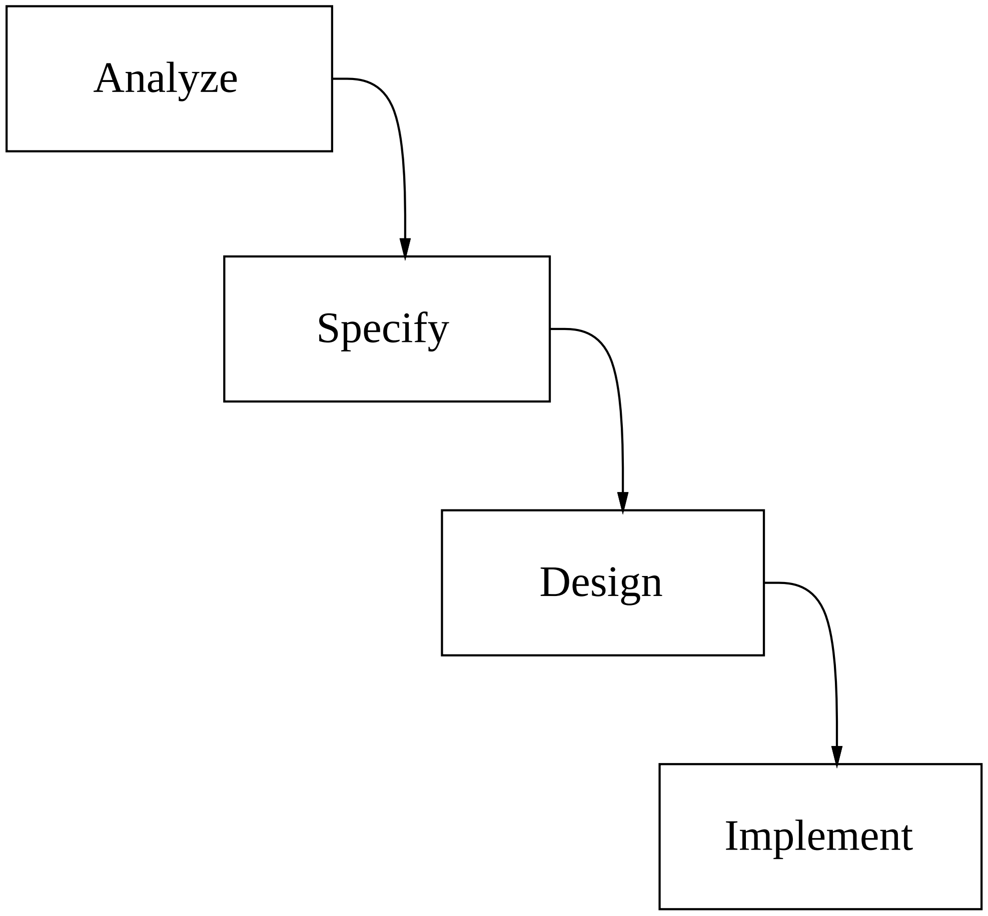


Figure 1: Major phases of SE process.

Process, cont'd

C. The **Analyze** step addresses requirements.

Process, cont’d

C. The **Analyze** step addresses requirements.

   1. Acquire and organize functional requirements of human users.

Process, cont’d

C. The **Analyze** step addresses requirements.

1. Acquire and organize functional require-
   ments of human users.

2. Involves considerable human-to-human
   communication.

Process, cont'd

D. The **Specify** step involves formal modeling of requirements.

Process, cont’d

D. The **Specify** step involves formal modeling of requirements.

   1. Model can be mechanically analyzed.

Process, cont'd

D. The **Specify** step involves formal modeling of requirements.

   1. Model can be mechanically analyzed.

   2. Checked for completeness and consistency.

Process, cont’d

E. The **Design** step involves organizing major software components.

Process, cont'd

E. The **Design** step involves organizing major software components.

   1. Initial design derived from spec model.

Process, cont'd

E. The **Design** step involves organizing major software components.

1. Initial design derived from spec model.

2. Refined into software architecture.

Process, cont’d

F. The **Implement** step fills in operational details.

Process, cont’d

F. The **Implement** step fills in operational details.

  1. Data structure details are determined.

Process, cont’d

F. The **Implement** step fills in operational details.

1. Data structure details are determined.
2. Code for methods is implemented.

Process, cont’d

G. Noteworthy process considerations.

Process, cont’d

G. Noteworthy process considerations.

1. "Ideally", steps completed in order.

Process, cont’d

G. Noteworthy process considerations.

1. "Ideally", steps completed in order.

   a. Figure 1 seen as a "waterfall chart".

Process, cont’d

G. Noteworthy process considerations.

1. "Ideally", steps completed in order.

   a. Figure 1 seen as a "waterfall chart".

   b. Information only flows down.

Process, cont’d

2. An "ideal" waterfall is rarely possible.

Process, cont’d

2. An "ideal" waterfall is rarely possible.
   a. Water sometimes flows up.

Process, cont’d

2. An "ideal" waterfall is rarely possible.

    a. Water sometimes flows up.

    b. Need feed-back from lower to higher steps.

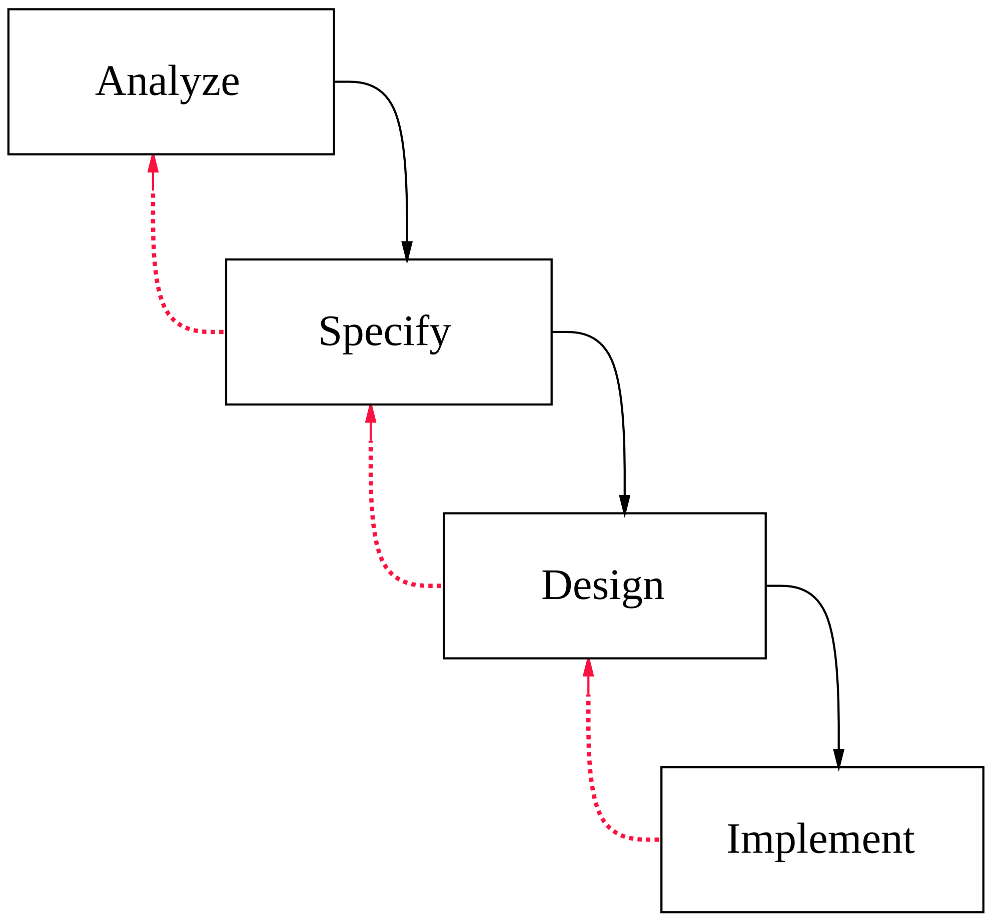


Figure 1: Updated SE process.

Process, cont’d

3. In the 307 process:

Process, cont’d

3. In the 307 process:

   a. Much feedback between **Analyze** & **Specify**

Process, cont'd

3. In the 307 process:

   a. Much feedback between **Analyze** & **Specify**

   b. Much feedback between **Design** & **Imple**

Process, cont’d

3. In the 307 process:

   a. Much feedback between **Analyze** & **Specify**

   b. Much feedback between **Design** & **Imple**

   c. Feedback from Design back up is limited.

Process, cont'd

H. Viewing process as problem solving:

Process, cont’d

H. Viewing process as problem solving:

1. Requirements & specification are
   problem statement

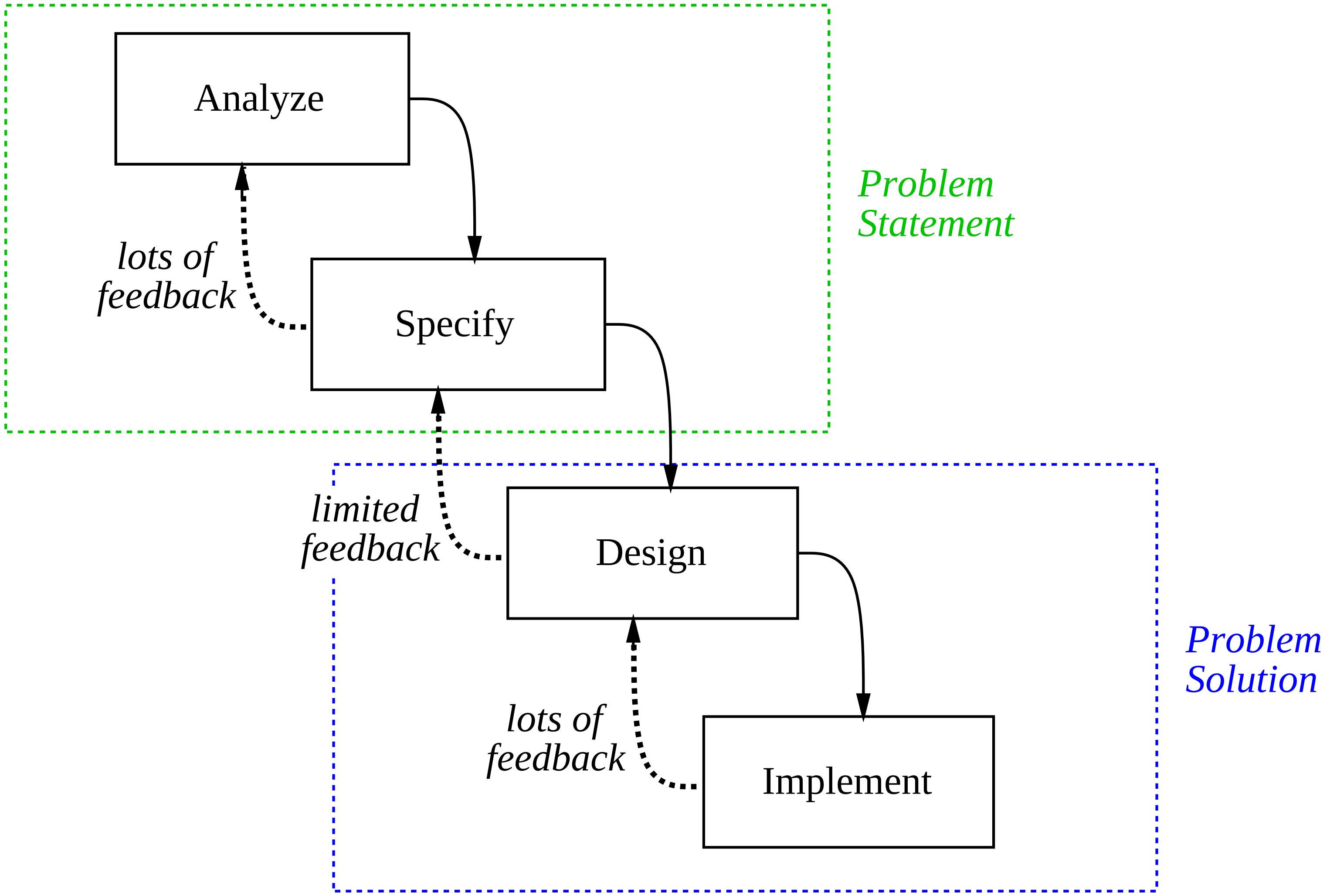
Process, cont'd

H. Viewing process as problem solving:

1. Requirements & specification are
   problem statement

2. Design & implementation are
   problem solution

Process as problem solving, cont'd



VII. Pervasive steps of the software process.

VII. **Pervasive steps of the software process.**

A. Figure 1 shows *ordered* process steps.

VII. **Pervasive steps of the software process.**

A. Figure 1 shows *ordered* process steps.

B. Even with feedback, overall order is

Analyze, **Specify**, **Design**, **Implement**.

VII. **Pervasive steps of the software process.**

A. Figure 1 shows *ordered* process steps.

B. Even with feedback, overall order is

Analyze, **Specify**, **Design**, **Implement**.

C. There are other steps that happen continuously, or "pervasively", throughout process:

Pervasive steps, cont’d

D. The pervasive steps of the process are:

Pervasive steps, cont’d

D. The pervasive steps of the process are:

1. **Manage**

Pervasive steps, cont'd

D. The pervasive steps of the process are:

1. **Manage**

2. **Configure**

Pervasive steps, cont’d

D. The pervasive steps of the process are:

1. **Manage**
2. **Configure**
3. **Test**

Pervasive steps, cont'd

D. The pervasive steps of the process are:

1. **Manage**
2. **Configure**
3. **Test**
4. **Document**

Pervasive steps, cont’d

D. The pervasive steps of the process are:

1. **Manage**
2. **Configure**
3. **Test**
4. **Document**
5. **Reuse**

Pervasive steps, cont'd

E. The **Manage** step entails management of people involved in the process.

Pervasive steps, cont'd

E. The **Manage** step entails management of people involved in the process.

1. Project meetings are scheduled at regular inter- vals.

Pervasive steps, cont'd

E. The **Manage** step entails management of people involved in the process.

1. Project meetings are scheduled at regular inter-vals.

2. Project supervisors oversee and evaluate the work of their subordinates.

Pervasive steps, cont'd

F. The **Configure** step entails organization and management of software artifacts.

Pervasive steps, cont'd

F. The **Configure** step entails organization and management of software artifacts.

1. Supported by version control tools.

Pervasive steps, cont’d

F. The **Configure** step entails organization and management of software artifacts.

1. Supported by version control tools.

2. The tools manage a software repository.

Pervasive steps, cont’d

G. The **Test** step ensures artifacts meet measurable standards.

Pervasive steps, cont’d

G. The **Test** step ensures artifacts meet measurable standards.

1. Testing requirements involves careful human inspection.

Pervasive steps, cont'd

G. The **Test** step ensures artifacts meet measurable standards.

1. Testing requirements involves careful human inspection.

2. Testing spec and design involves formal analysis.

Pervasive steps, cont’d

G. The **Test** step ensures artifacts meet measurable standards.

1. Testing requirements involves careful human inspection.

2. Testing spec and design involves formal analysis.

3. Testing implementation involves formal functional testing.

Pervasive steps, cont'd

H. The **Document** step produces documents suitable for everyone involved.

Pervasive steps, cont'd

H. The **Document** step produces documents suit-able for everyone involved.

1. Requirements spec document.

Pervasive steps, cont'd

H. The **Document** step produces documents suit-able for everyone involved.

1. Requirements spec document.

2. Maintenance documentation.

Pervasive steps, cont'd

H. The **Document** step produces documents suit-able for everyone involved.

1. Requirements spec document.
2. Maintenance documentation.
3. Project reports.

Pervasive steps, cont’d

H. The **Document** step produces documents suitable for everyone involved.

1. Requirements spec document.
2. Maintenance documentation.
3. Project reports.
4. End user manuals and tutorials.

Pervasive steps, cont'd

I. The **Reuse** step evaluates existing artifacts to determine if they can be reused.

Pervasive steps, cont’d

I. The **Reuse** step evaluates existing artifacts to determine if they can be reused.

1. Reuse from libraries is normal.

Pervasive steps, cont’d

I. The **Reuse** step evaluates existing artifacts to determine if they can be reused.

1. Reuse from libraries is normal.

2. Reuse of other artifacts involves refining and adapting.

Pervasive steps, cont’d

J. Important characteristics of pervasive steps.

Pervasive steps, cont’d

J. Important characteristics of pervasive steps:

1. May be performed *during* ordered steps.

Pervasive steps, cont'd

J. Important characteristics of pervasive steps.

1. May be performed *during* ordered steps.

2. May be regularly scheduled.

VIII. **Traditional process versus agile processes.**

VIII. **Traditional process versus agile processes.**

A. 307 process considered *traditional*.

VIII. **Traditional process versus agile processes.**

A. 307 process considered *traditional*.

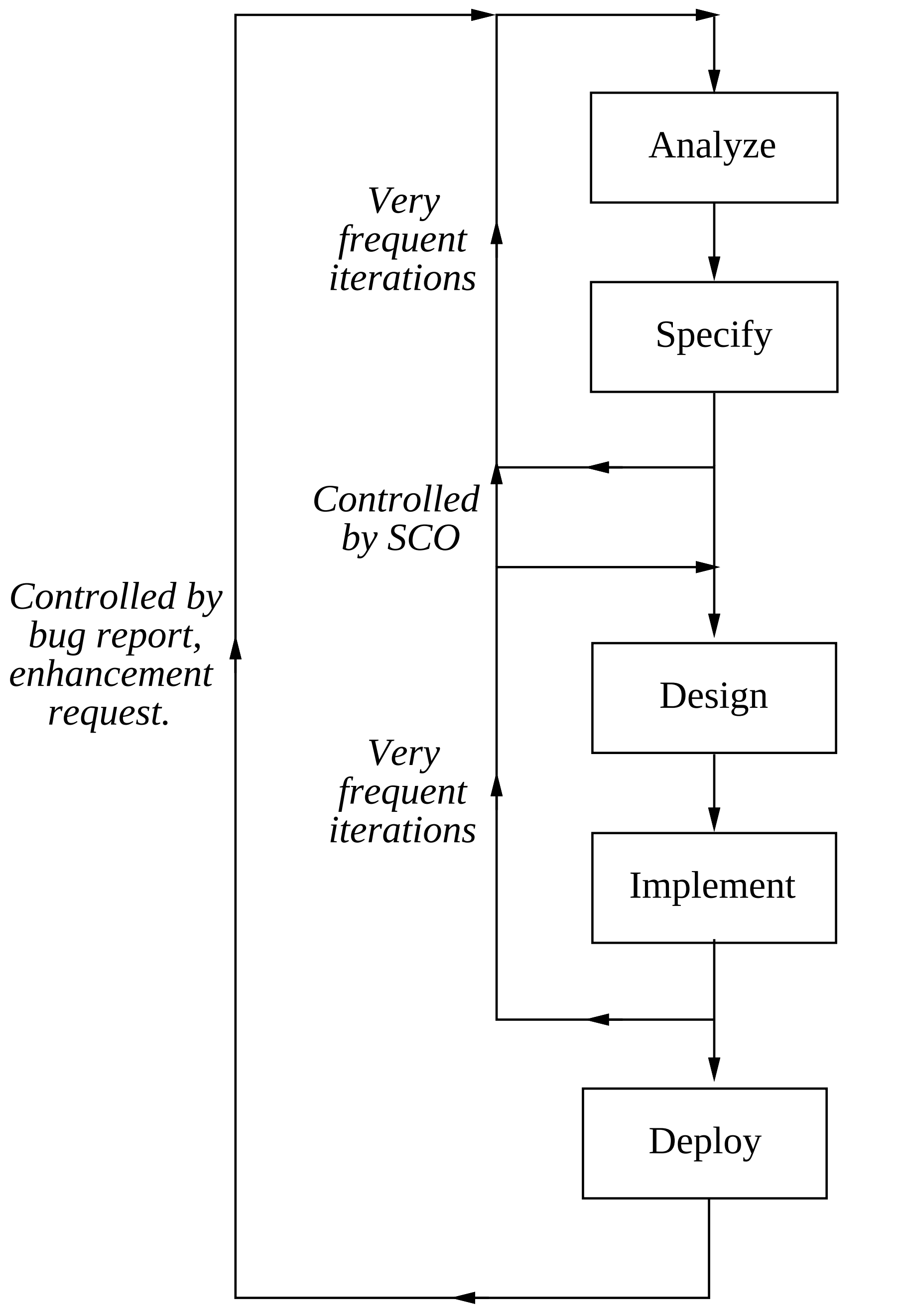
B. Particularly the production of a substantial requirements document.

VIII. **Traditional process versus agile processes.**

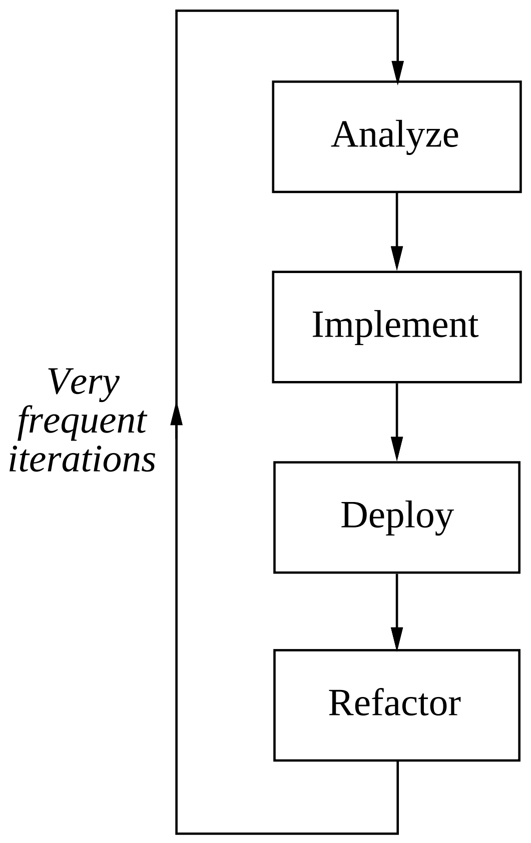
A. 307 process considered *traditional*.

B. Particularly the production of a substantial requirements document.

C. More incremental is *agile development*.



a. Traditional process



b. Agile process

Traditional versus agile, cont'd

D. In agile development, or
extreme programming:

Traditional versus agile, cont’d

D. In agile development, or *extreme programming*:

1. Customers and implementors work very closely together.

Traditional versus agile, cont’d

D. In agile development, or
extreme programming:

1. Customers and implementors work very closely together.

2. Traditional steps of **specification** & **design** replaced by "refactoring".

IX. Details of Analyze and Specify Steps

IX. Details of Analyze and Specify Steps

A. Precisely specify need.

IX. **Details of Analyze and Specify Steps**

A. Precisely specify need.

B. In a requirements specification document.

IX. Details of Analyze and Specify Steps

A. Precisely specify need.

B. In a requirements specification document.

C. Informal sections of document are *understandable to everyone.*

IX. **Details of Analyze and Specify Steps**

A. Precisely specify need.

B. In a requirements specification document.

C. Informal sections of document are *understandable to everyone.*

D. Formal sections precise enough to be a *contractual instrument.*

X. Importance of careful analysis.

X. Importance of careful analysis.

A. We must have a precise understanding of exactly what user needs are.

X. Importance of careful analysis.

A. We must have a precise understanding of exactly what user needs are.

B. A seemingly obvious idea.

X. **Importance of careful analysis.**

A. We must have a precise understanding of exactly what user needs are.

B. A seemingly obvious idea.

C. Lure of technology may lead to insufficient time spent on requirements.

Importance of analysis, cont’d

D. Hastily-acquired software can cause problems.

Importance of analysis, cont’d

D. Hastily-acquired software can cause problems.

E. Hastily-marketed software may not succeed.

Importance of analysis, cont’d

E. Hastily-acquired systems can cause problems.

F. Hastily-marketed software may not succeed.

G. There’s universal agreement that understanding requirements is absolutely necessary.

XI. Patience is required.

XI. **Patience is required.**

A. Things may seem obvious.

XI. **Patience is required.**

A. Things may seem obvious.

B. Many think they have a clear idea.

XI. **Patience is required.**

A. Things may seem obvious.

B. Many think they have a clear idea.

C. Everyone may not have *same* idea.

XI. **Patience is required.**

A. Things may seem obvious.

B. Many think they have a clear idea.

C. Everyone may not have *same* idea.

D. Precise analysis helps everyone agree.

XII. Major phases of requirements specification

XII. **Major phases of requirements specification**

A. End-user scenarios.

XII. **Major phases of requirements specification**

A. End-user scenarios.

1. Language used is English and pictures.

XII. **Major phases of requirements specification**

A. End-user scenarios.

1. Language used is English and pictures.

2. Primary audience is customers, end users.

XII. **Major phases of requirements specification**

A. End-user scenarios.

1. Language used is English and pictures.
2. Primary audience is customers, end users.
3. Much user consultation required.

Major phases, cont’d

B. Formal model specification.

Major phases, cont’d

B. Formal model specification.

   1. Formal spec language is used.

Major phases, cont’d

B. Formal model specification.

1. Formal spec language is used.
2. Primary audience is development team.

Major phases, cont’d

B. Formal model specification.

1. Formal spec language is used.
2. Primary audience is development team.
3. Final version is a *very* formal.

XIII. Details of user consultations

XIII. Details of user consultations

A. Critically important to involve end users.

XIII. **Details of user consultations**

A. Critically important to involve end users.

B. Success is far more likely.

XIII. **Details of user consultations**

A. Critically important to involve end users.

B. Success is far more likely.

C. Many serious failures have resulted when end users are neglected.

XIV. Activities of user consultation

XIV. **Activities of user consultation**

A. User interviews.

XIV. **Activities of user consultation**

A. User interviews.

B. User interface scenarios.

XIV. **Activities of user consultation**

A. User interviews.

B. User interface scenarios.

C. User questionnaires or surveys.

XIV. **Activities of user consultation**

A. User interviews.

B. User interface scenarios.

C. User questionnaires or surveys.

D. Visits to other similar installations.

XIV. **Activities of user consultation**

A. User interviews.

B. User interface scenarios.

C. User questionnaires or surveys.

D. Visits to other similar installations.

E. Rapid system prototypes.

XV. Interview techniques

XV. **Interview techniques**

A. Minimize computer jargon.

XV. **Interview techniques**

A. Minimize computer jargon.

B. Specialize questions to each user.

XV. **Interview techniques**

A. Minimize computer jargon.

B. Specialize questions to each user.

C. Use common sense -- be prepared, polite, succinct, non-threatening, diplomatic, empathetic.

XVI. User interface scenarios

XVI. User interface scenarios

A. Provide users with a concrete view.

XVI. **User interface scenarios**

A. Provide users with a concrete view.

B. Premise: *"Suppose the system existed already, what would it look like?"*

XVI. **User interface scenarios**

A. Provide users with a concrete view.

B. Premise: *"Suppose the system existed already, what would it look like?"*

1. Define precisely what user sees.

XVI. **User interface scenarios**

A. Provide users with a concrete view.

B. Premise: *"Suppose the system existed already, what would it look like?"*

1. Define precisely what user sees.

2. Screens, commands, data formats, and all other user-visible aspects of operation.

XVII. (Rapid) Prototyping

XVII. (Rapid) Prototyping

A. Helps capture user requirements.

XVII. **(Rapid) Prototyping**

A. Helps capture user requirements.

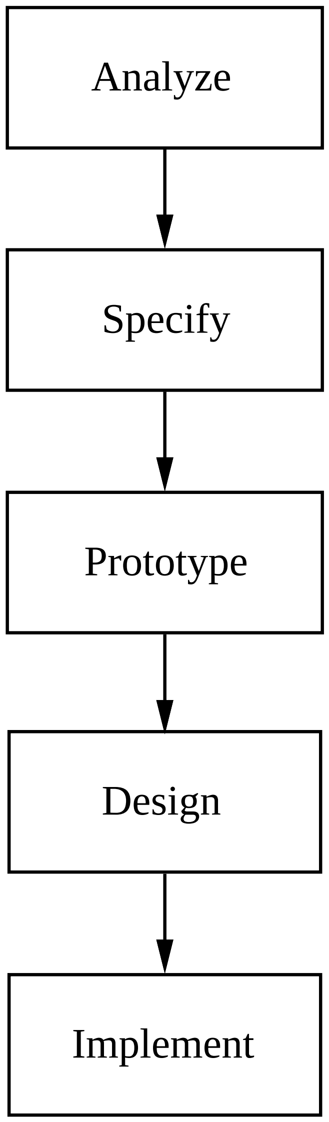
B. Version of software with reduced functionality.

XVII. (Rapid) Prototyping

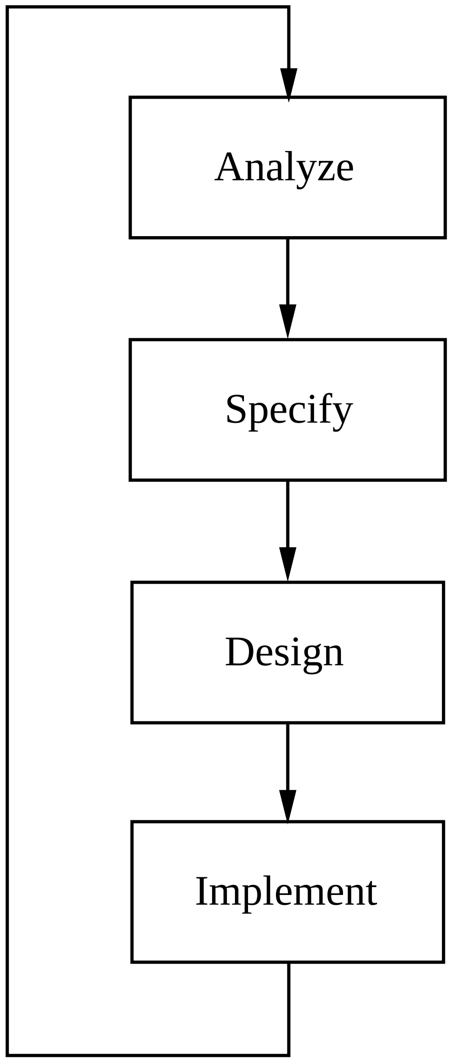
A. Helps capture user requirements.

B. Version of software with reduced functionality.

C. Figure 2 shows two views or prototyping.



a. As explicit process step



b. As multiple passes

Prototyping, cont'd

D. In 307, we'll Fig 2a, with a GUI prototype before the detailed software design.

XVIII. Establishing genuine user needs

XVIII. **Establishing genuine user needs**

A. One more time -- ***it's critical.***

XVIII. **Establishing genuine user needs**

A. One more time -- ***it's critical.***

B. Need for software must be clear.

XVIII. **Establishing genuine user needs**

A. One more time -- ***it's critical.***

B. Need for software must be clear.

C. Once needs are established, software may be purchased or developed.

XIX. Other important aspects

XIX. **Other important aspects**

A. Identification of stakeholders.

XIX. **Other important aspects**

A. Identification of stakeholders.

B. Analysis of current and proposed operations.

XIX. **Other important aspects**

A. Identification of stakeholders.

B. Analysis of current and proposed operations.

C. Impact analysis.

XIX. **Other important aspects**

A. Identification of stakeholders.

B. Analysis of current and proposed operations.

C. Impact analysis.

D. Analysis of relevant existing systems.

XIX. **Other important aspects**

A. Identification of stakeholders.

B. Analysis of current and proposed operations.

C. Impact analysis.

D. Analysis of relevant existing systems.

E. ***These are in our requirements Section 1.***

XX. Examples of requirements specification

XX. **Examples of requirements specification**

A. Concrete examples similar in size and scope to your 307 project.

XX. **Examples of requirements specification**

A. Concrete examples similar in size and scope to your 307 project.

B. Examples for Milestones 1, 2, 4, 6, 8, and 10.

XX. **Examples of requirements specification**

A. Concrete examples similar in size and scope to your 307 project.

B. Examples for Milestones 1, 2, 4, 6, 8, and 10.

C. We’ll go over throughout the quarter.

Note:

During the first two weeks of in-class presentations, the handout slides that follow from here were intermingled with the preceding slides from the lecture notes "proper".

Milestone 1

Milestone 1

- Due Wed second week, check in by 11:59PM

Milestone 1

- Due Wed second week, check in by 11:59PM
- Tasks:

Milestone 1

- Due Wed second week, check in by 11:59PM
- Tasks:
  a. Refinements to M0 deliverables

Milestone 1

- Due Wed second week, check in by 11:59PM
- Tasks:
  a. Refinements to M0 deliverables
  b. Questions for week 2 customer interview

Milestone 1

- Due Wed second week, check in by 11:59PM
- Tasks:
  a. Refinements to M0 deliverables
  b. Questions for week 2 customer interview
  c. Rough draft of Section 1

Section 1: Introduction

Section 1: Introduction

- Initial paragraphs are executive summary.

Section 1: Introduction

- Initial paragraphs are executive summary.
- Use present tense, third person, active voice.

Section 1: Introduction

- Initial paragraphs are executive summary.
- Use present tense, third person, active voice.
- Use Calendar Tool example as overall guide.

Section 1.1: Problem Statement

Section 1.1: Problem Statement

- Succinct presentation of problem(s) to be solved.

Section 1.1: Problem Statement

- Succinct presentation of problem(s) to be solved.

- You may (or may not) include the problem of pro- viding a pedagogical example.

Section 1.2: System Personnel

Section 1.2: System Personnel

- Description of all people involved.

Section 1.2: System Personnel

- Description of all people involved.
- For M1, focus on end user categories.

Section 1.2: System Personnel

- Description of all people involved.
- For M1, focus on end user categories.
- E.g., for Calendar Tool categories are:

Section 1.2: System Personnel

- Description of all people involved.
- For M1, focus on end user categories.
- E.g., for Calendar Tool categories are:
    - registered users

Section 1.2: System Personnel

- Description of all people involved.
- For M1, focus on end user categories.
- E.g., for Calendar Tool categories are:
    - registered users
    - group leaders

Section 1.2: System Personnel

- Description of all people involved.
- For M1, focus on end user categories.
- E.g., for Calendar Tool categories are:
  - registered users
  - group leaders
  - master admins

Section 1.2: System Personnel

- Description of all people involved.
- For M1, focus on end user categories.
- E.g., for Calendar Tool categories are:
    - registered users
    - group leaders
    - master admins
    - unregistered users

Section 1.3: Operational Setting

Section 1.3: Operational Setting

- Environment in which tool is used.

Section 1.3: Operational Setting

- Environment in which tool is used.
- Describe before and after proposed system is installed.

Section 1.3: Operational Setting

- Environment in which tool is used.
- Describe before and after proposed system is installed.
- Consider if proposed system must interface with existing systems.

Section 1.4: Impact Analysis

Section 1.4: Impact Analysis

- Positive, negative impacts in proposed setting.

Section 1.4: Impact Analysis

- Positive, negative impacts in proposed setting.
- E.g., for Calendar Tool:

Section 1.4: Impact Analysis

- Positive, negative impacts in proposed setting.
- E.g., for Calendar Tool:
  - *Positive:* increased convenience and efficiency.

Section 1.4: Impact Analysis

- Positive, negative impacts in proposed setting.

- E.g., for Calendar Tool:

    - *Positive:* increased convenience and efficiency.

    - *Negative:* decreased privacy, potential disruption of business.

Section 1.5: Related Systems

Section 1.5: Related Systems

- Other software with similar functionality.

Section 1.5: Related Systems

- Other software with similar functionality.

- Consider:

Section 1.5: Related Systems

- Other software with similar functionality.
- Consider:
    - What is good about them.

Section 1.5: Related Systems

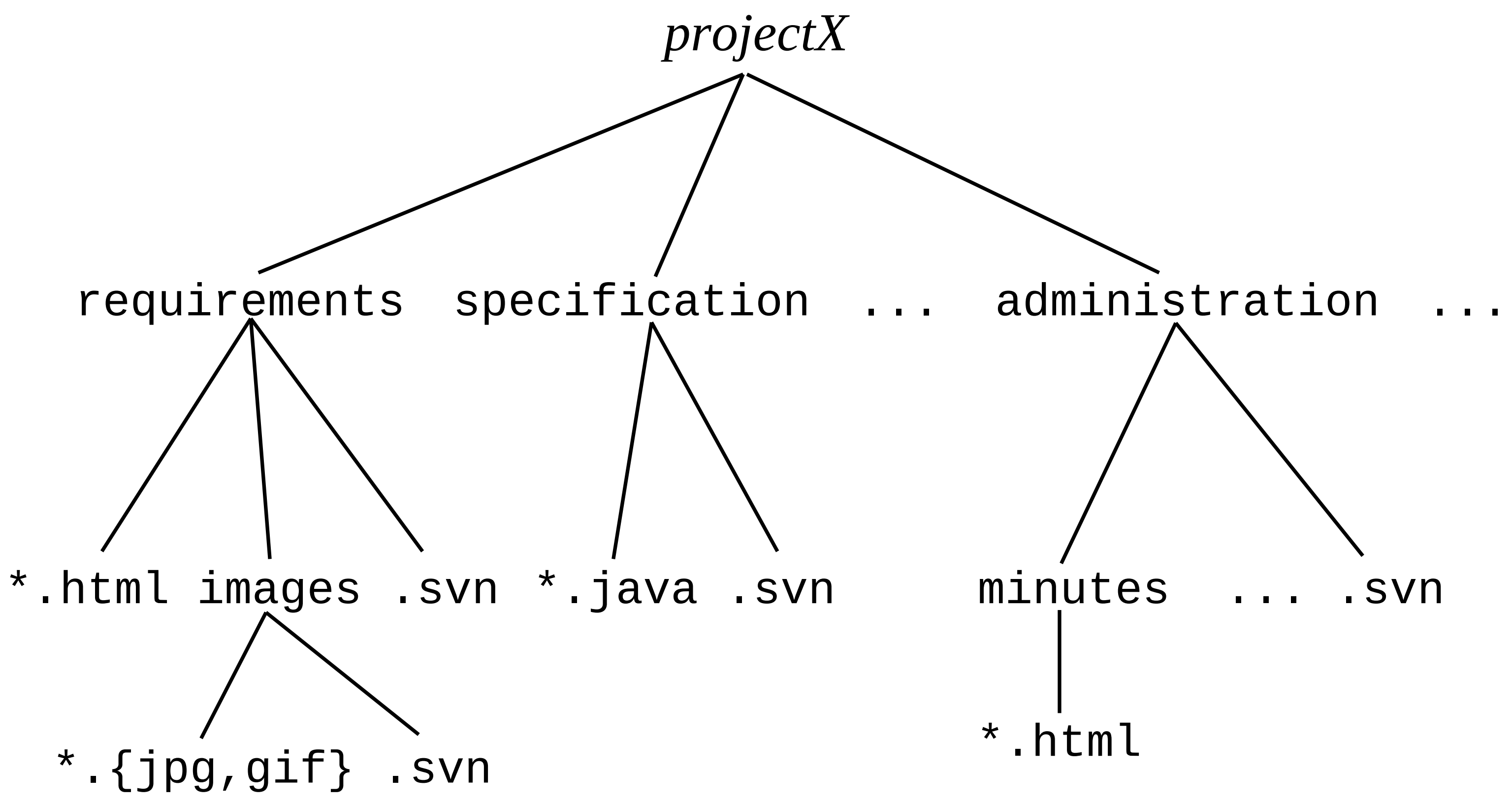
- Other software with similar functionality.
- Consider:
    - What is good about them.
    - What is bad.

Section 1.5: Related Systems

- Other software with similar functionality.
- Consider:
  - What is good about them.
  - What is bad.
  - What is missing.

SOP Volume 1

Project Directory Structure



Specific Update Procedures

Specific Update Procedures

- Each project member (including librarian) has her/his own *work* directory.

Specific Update Procedures

- Each project member (including librarian) has her/his own *work* directory.

- There is a master *projects* directory maintained by the project librarian.

Specific Update Procedures

- Each project member (including librarian) has her/his own *work* directory.

- There is a master *projects* directory maintained by the project librarian.

- See Figure 2 in handout.

Update Procedures, cont’d

Update Procedures, cont'd

- Changes originate in individual work directories.

Update Procedures, cont’d

- Changes originate in individual work directories.

- Team members checkin their work using
  svn add and *svn commit*.

Update Procedures, cont’d

- Changes originate in individual work directories.
- Team members checkin their work using
  svn add and *svn commit*.
- Team members checkout colleagues’ work using
  svn update.

Update Procedures, cont'd

- Changes originate in individual work directories.
- Team members checkin their work using *svn add* and *svn commit*.
- Team members checkout colleagues' work using *svn update*.
- Librarian releases to project directory using *svn update*.

Update Procedures, cont’d

Update Procedures, cont’d

- Check in happens at least weekly.

Update Procedures, cont’d

- Check in happens at least weekly.
- Individuals check in their work.

Update Procedures, cont’d

- Check in happens at least weekly.
- Individuals check in their work.
- Librarian "releases" to public project directory.

File Ownership

File Ownership

- Exactly one member owns each file.

File Ownership

- Exactly one member owns each file.
- Owner has check in authority.

File Ownership

- Exactly one member owns each file.
- Owner has check in authority.
- Other members check out at will.

File Ownership

- Exactly one member owns each file.
- Owner has check in authority.
- Other members check out at will.
- Ownership recorded in file

```
    administration/
         work-breakdown.html
```

SVN Basics

SVN Basics

- SVN is "Subversion" version control tool.

SVN Basics

- SVN is "Subversion" version control tool.

- It maintains a version *repository* that records the history of a project’s files.

SVN Basics

- SVN is "Subversion" version control tool.

- It maintains a version *repository* that records the history of a project’s files.

- Members of a project team each maintain an individual *working* directory.

SVN Basics, cont’d

SVN Basics, cont'd

- There are two fundamental operations of any version control system:

SVN Basics, cont’d

- There are two fundamental operations of any version control system:

  - file *check in*, from a individual working directory to the repository

SVN Basics, cont’d

- There are two fundamental operations of any version control system:

  - file *check in*, from a individual working directory to the repository

  - file *check out*, from the repository to a working directory

SVN Basics, cont’d

SVN Basics, cont’d

- In SVN, check in is accomplished using the *svn add* and *svn commit* commands.

SVN Basics, cont’d

- In SVN, check in is accomplished using the *svn add* and *svn commit* commands.

- Check out is done most frequently with the *svn update* command.

SVN Basics, cont’d

SVN Basics, cont'd

- Other useful SVN commands include:

SVN Basics, cont’d

- Other useful SVN commands include:
  - removing unnecessary files

SVN Basics, cont'd

- Other useful SVN commands include:
  - *o* removing unnecessary files
  - *o* checking file status

SVN Basics, cont'd

- Other useful SVN commands include:
  - *o* removing unnecessary files
  - *o* checking file status
  - *o* controlling which files are put in repository

SVN Basics, cont’d

- Other useful SVN commands include:
  - removing unnecessary files
  - checking file status
  - controlling which files are put in repository
  - comparing past versions

SVN Basics, cont’d

- Other useful SVN commands include:
  - *o* removing unnecessary files
  - *o* checking file status
  - *o* controlling which files are put in repository
  - *o* comparing past versions

- SVN basics handout covers details.

SVN Basics, cont’d

1. **Initial library setup**

   Done by librarian one time only.

SVN Basics, cont’d

2. Initial project checkout

```
cd
mkdir work
cd work
svn checkout file:///home/librarian/
   your-project/projects/SVN/trunk/your-project
```

Performed one time only.

SVN Basics, cont'd

3. Checkin new work

```
cd ~/work/your-project/...
create some-file
svn add some-file
svn commit -m "log message"  some-file
```

Performed the first time you check in a file.

SVN Basics, cont’d

4. Checkin revised work

`cd ~/work/`*your-project/*`...`
edit some-file
`svn commit -m "`*log message*`"` *some-file*

Performed every time you revise a file.

SVN Basics, cont’d

5. Checkout team members’ work

```
cd ~/work/your-project
svn update
```

Performed to get your teammates’ latest work.

SVN Basics, cont'd

6. Release (by librarian) of team work

```
cd ~librarian/projects/work/your-project
svn update
```

Performed by librarian to hand in group's work.

SVN Basics, cont’d

7. Removing previous checked in files

To remove file named "*X*" from repository:

```
svn remove -f X
svn commit -m "log message"
```

Performed to remove a file from the repository.

SVN Basics, cont’d

8. Viewing status

```
cd ~/work/your-project
svn status -u
```

Produces file list with the following status codes:

SVN Basics, cont’d

Code	Meaning
M	**M**odified file, i.e., you’ve made some changes and need to commit the file.
?	Unknown file, need to add and commit it.
!	UNIX rm’d file wihtout svn remove.

SVN Basics, cont’d

Code	Meaning
A	**A**dded file via ‘`svn add`’, needs to be committed.
R	**R**emoved file via ‘`svn remove`’, needs to be committed.
C	**C**onflict exists (see below for details).

SVN Basics, cont'd

- If '`*`' appears, team member has made changes.
- If both '`M`' and '`*`', conflict exists -- see below.

SVN Basics, cont'd

9. Differencing Modified Files

For any file *X*,

```
svn diff X
```

diffs working and repository copies.

SVN Basics, cont'd

10. Viewing a log report

For any file *X*,

```
svn log X
```

or for an entire directory recursively, just

```
svn log
```

SVN Basics, cont’d

11. Undoing Working Changes

For added or removed file *X*,

```
svn revert X
```

undoes add or remove.

Also erases local uncommitted changes.

SVN Basics, cont’d

12. Dealing with a Conflict

For conflicting file `X`,

```
mv X X.sav
svn update X
```

Then compare `X` with `X.sav` to see how to deal with the differences.

SVN Basics, cont’d

13. Telling svn to ignore certain files

In the directory where the files to be ignored reside, add file names into `.svnignore` file. Then

```
svn propset svn:ignore -F .svnignore .
svn commit -m "Ignored files ..."
```

SVN Basics, cont'd

14. Connecting to a SVN server remotely

- Install `svn` and `ssh`, if necessary.
- Run

```
svn checkout svn+ssh://id@unix3/home/librarian/
   your-project/projects/SVN/trunk/your-project
```

- Use command line or GUI client.
- See Lab Notes 3 for more details.

Milestone 2 Writeup

Milestone 2 Writeup

- Due Wed third week

Milestone 2 Writeup

- Due Wed third week
- Activities:

Milestone 2 Writeup

- Due Wed third week

- Activities:
  - Initial rough draft of Section 2.

Milestone 2 Writeup

- Due Wed third week

- Activities:
  - Initial rough draft of Section 2.
  - Top-Level UI(s).

Milestone 2 Writeup

- Due Wed third week

- Activities:
  - Initial rough draft of Section 2.
  - Top-Level UI(s).
  - Draft table of contents.

Milestone 2 Writeup

- Due Wed third week

- Activities:
  - Initial rough draft of Section 2.
  - Top-Level UI(s).
  - Draft table of contents.
  - Two scenarios per team member,
    minimum two distinct screens per member.

Milestone 2 Writeup

- Due Wed third week

- Activities:
  - Initial rough draft of Section 2.
  - Top-Level UI(s).
  - Draft table of contents.
  - Two scenarios per team member, ***minimum two distinct screens per member.***
  - Update `admin/work-breakdown.html`

What’s a Scenario?

What’s a Scenario?

- Describes some aspect of using a program.

What’s a Scenario?

- Describes some aspect of using a program.
- A scenario has a sequence of ***use cases***.

What’s a Scenario?

- Describes some aspect of using a program.
- A scenario has a sequence of ***use cases***.
- Each use case describes:

What’s a Scenario?

- Describes some aspect of using a program.
- A scenario has a sequence of ***use cases***.
- Each use case describes:
  - a specific user action

What's a Scenario?

- Describes some aspect of using a program.
- A scenario has a sequence of ***use cases***.
- Each use case describes:
  - a specific user action
  - the program's response

What's a Scenario?

- Describes some aspect of using a program.
- A scenario has a sequence of ***use cases***.
- Each use case describes:
  - a specific user action
  - the program's response
  - a detailed description of the response

Rough Draft Scenarios in Milestone 2

Rough Draft Scenarios in Milestone 2

- Lay out main TestTool UI and describe it.

Rough Draft Scenarios in Milestone 2

- Lay out main TestTool UI and describe it.
- Describe what happens when user does some-thing, like pressing a button or menu item.

Rough Draft Scenarios in Milestone 2

- Lay out main TestTool UI and describe it.
- Describe what happens when user does something, like pressing a button or menu item.
- Show next level UIs and describe each of them.

Rough Draft Scenarios in Milestone 2

- Lay out main TestTool UI and describe it.
- Describe what happens when user does something, like pressing a button or menu item.
- Show next level UIs and describe each of them.
- Keep going like this.

Rough Draft Scenarios in Milestone 2

- Lay out main TestTool UI and describe it.
- Describe what happens when user does something, like pressing a button or menu item.
- Show next level UIs and describe each of them.
- Keep going like this.
- The Milestone 2 example illustrates.

Milestone 2 Example

Milestone 2 Example

- Very rough draft of requirements.

Milestone 2 Example

- Very rough draft of requirements.
- Sections 1 and 2 of requirements doc.

Milestone 2 Example

- Very rough draft of requirements.
- Sections 1 and 2 of requirements doc.
- Calendar project is similar to yours.

Milestone 2 Example

- Very rough draft of requirements.
- Sections 1 and 2 of requirements doc.
- Calendar project is similar to yours.
- Editorial notes provide explanation.

Milestone 2 Example

- Very rough draft of requirements.
- Sections 1 and 2 of requirements doc.
- Calendar project is similar to yours.
- Editorial notes provide explanation.
- For M2, focus on content primarily.

Section 2: Functional Requirements

Section 2: Functional Requirements

- Definition of all functions and data.

Section 2: Functional Requirements

- Definition of all functions and data.
- In scenarios depicting end-user interactions.

Section 2: Functional Requirements

- Definition of all functions and data.
- In scenarios depicting end-user interactions.
- Scenarios are in tutorial style.

Section 2: Functional Requirements

- Definition of all functions and data.
- In scenarios depicting end-user interactions.
- Scenarios are in tutorial style.
  - Tell interesting and engaging story.

Section 2: Functional Requirements

- Definition of all functions and data.
- In scenarios depicting end-user interactions.
- Scenarios are in tutorial style.
  - Tell interesting and engaging story.
  - Give step-by-step presentation.

Section 2: Functional Requirements

- Definition of all functions and data.
- In scenarios depicting end-user interactions.
- Scenarios are in tutorial style.
  - Tell interesting and engaging story.
  - Give step-by-step presentation.
  - Eventually cover all functionality.

Section 2.1: User-Interface Overview

Section 2.1: User-Interface Overview

- Standard section for all projects.

Section 2.1: User-Interface Overview

- Standard section for all projects.
- Present functional hierarchy of tool operations.

Section 2.1: User-Interface Overview

- Standard section for all projects.
- Present functional hierarchy of tool operations.
- Example uses menubar as concrete representation; ***you need not*,** but must have equivalent.

UI Overview, cont’d

- Note use of *very simple* GUI.

UI Overview, cont'd

- Note use of *very simple* GUI.

- More on GUI conventions in handout.

UI Overview, cont’d

- Note use of *very simple* GUI.
- More on GUI conventions in handout.
- ***IMPORTANT:*** Do not get bogged down in low-level GUI details in early stages of requirements.

UI Overview, cont’d

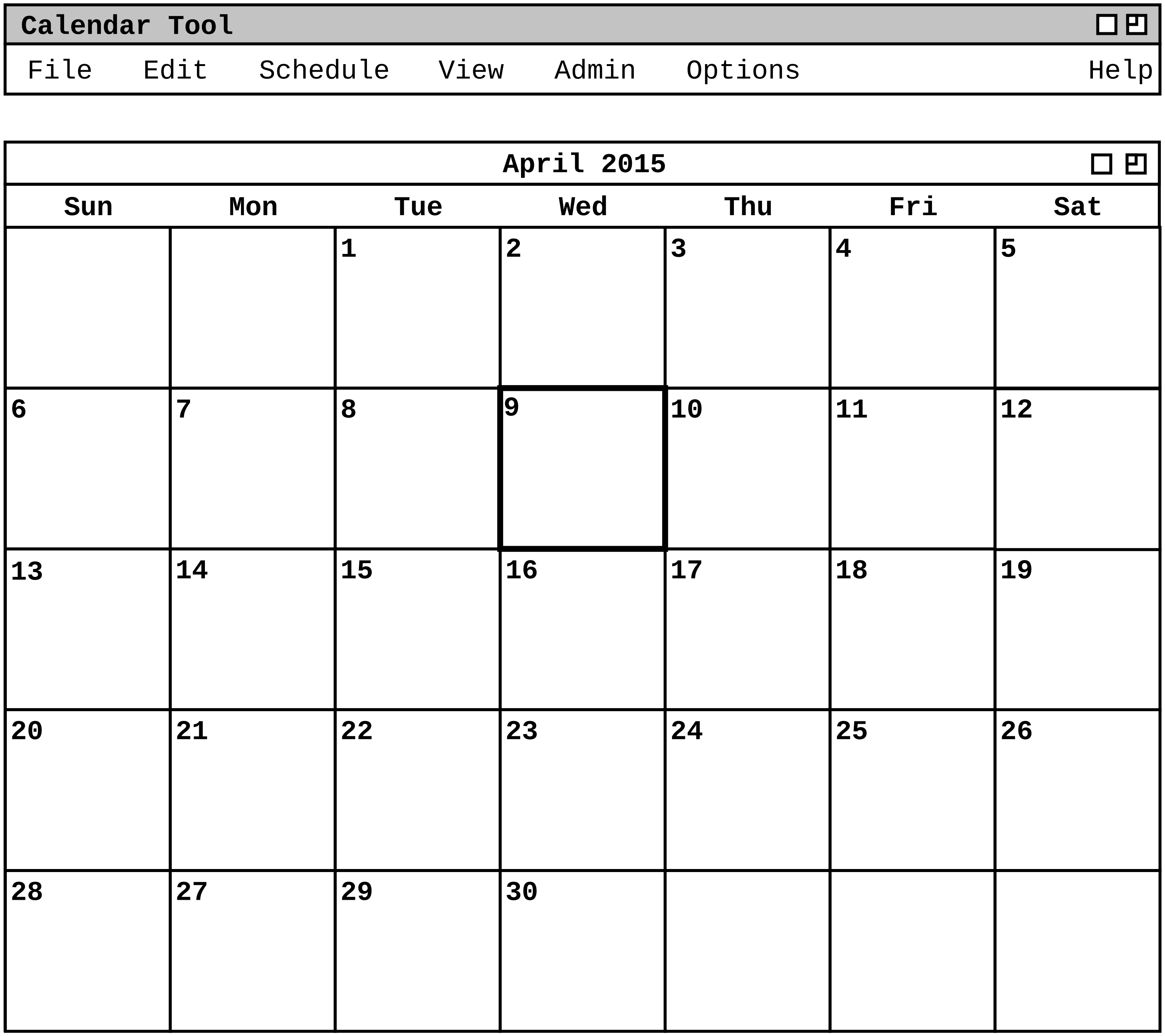
- Start with "When the user initially invokes ..."

UI Overview, cont’d

- Start with "When the user initially invokes ..."
- Figure 1 shows initial default screen.

UI Overview, cont’d

- Start with "When the user initially invokes ..."
- Figure 1 shows initial default screen.
- E.g., here’s Figure 1 for Calendar example:



April 2015

Sun	Mon	Tue	Wed	Thu	Fri	Sat
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
28	27	29	30			

UI Overview, cont’d

- How system starts "out of the box" for typical user.

UI Overview, cont’d

- How system starts "out of the box" for typical user.

- Prose narrative following screen explains content.

UI Overview, cont'd

- Figure 2 shows expansion of command menus.

UI Overview, cont’d

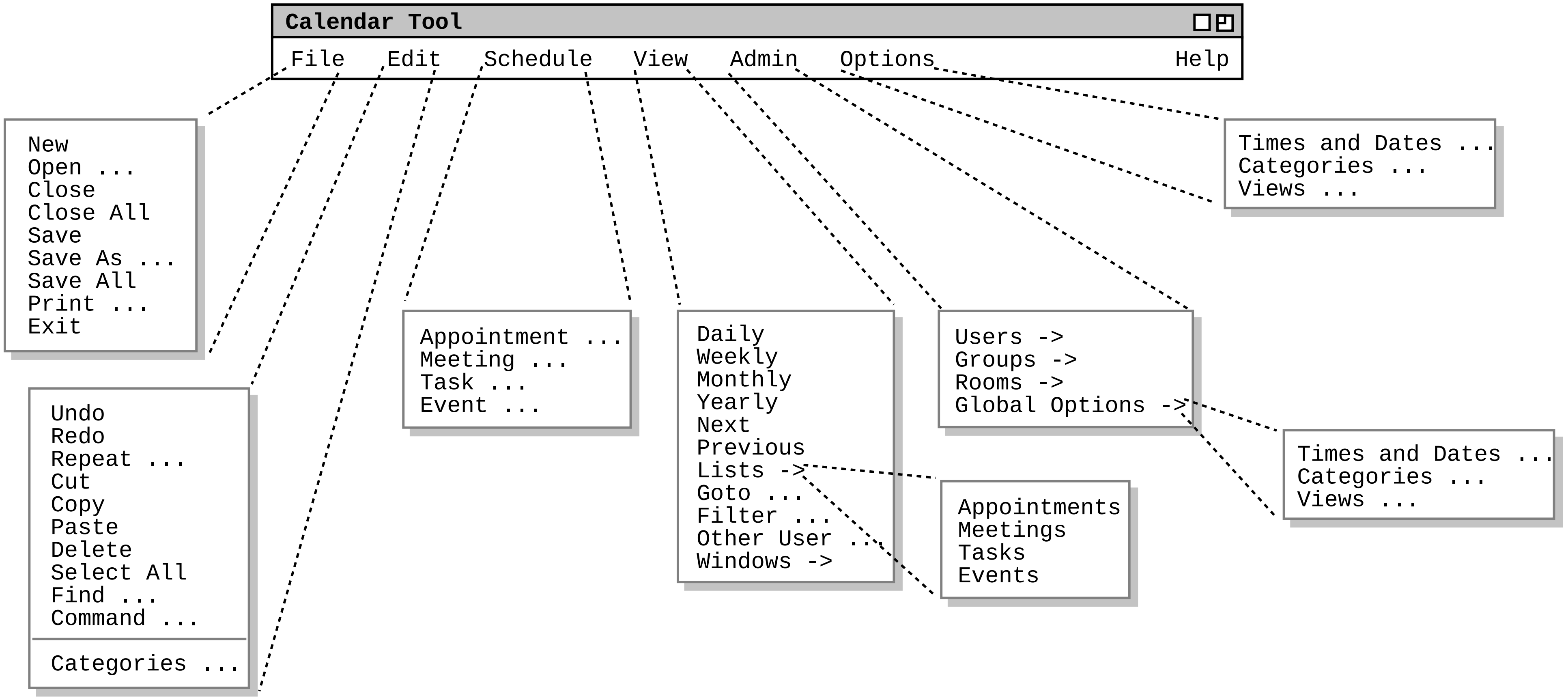
- Figure 2 shows expansion of command menus.
- Concrete representation of pulldown menu is convenient standard format.

UI Overview, cont’d

- Figure 2 shows expansion of command menus.
- Concrete representation of pulldown menu is convenient standard format.
- Conceptually, we are presenting a
  functional command hierarchy.

UI Overview, cont’d

- Figure 2 shows expansion of command menus.
- Concrete representation of pulldown menu is convenient standard format.
- Conceptually, we are presenting a
  functional command hierarchy.
- E.g., here’s Figure 2 for Calendar example:


Calendar Tool
File
Edit
Schedule
View
Admin
Options
Help
New
Open ...
Close
Close All
Save
Save As ...
Save All
Print ...
Exit
Undo
Redo
Repeat ...
Cut
Copy
Paste
Delete
Select All
Find ...
Command ...
Categories ...
Appointment ...
Meeting ...
Task ...
Event ...
Daily
Weekly
Monthly
Yearly
Next
Previous
Lists ->
Goto ...
Filter ...
Other User ...
Windows ->
Appointments
Meetings
Tasks
Events
Users ->
Groups ->
Rooms ->
Global Options ->
Times and Dates ...
Categories ...
Views ...
Times and Dates ...
Categories ...
Views ...

UI Overview, cont’d

- A pulldown menu is not the only way to represent a functional command hierarchy.

UI Overview, cont’d

- A pulldown menu is not the only way to represent a functional command hierarchy.

- It’s a widely-recognized UI standard, at present.

UI Overview, cont’d

- A pulldown menu is not the only way to represent a functional command hierarchy.
- It’s a widely-recognized UI standard, at present.
- Next slide shows equivalent functional hierarchy in plain text form.

UI Overview, cont’d

- A pulldown menu is not the only way to represent a functional command hierarchy.
- It’s a widely-recognized UI standard, at present.
- Next slide shows equivalent functional hierarchy in plain text form.
- Plain text form is acceptable for Milestone 2.

File:

- New
- Open
- Close
- Close All
- Save
- Save As
- Save All
- Print
- Exit

Edit:

- Undo
- Redo
- Repeat
- Cut
- Copy
- Paste
- Delete
- Select All
- Find
- Command
- Categories

Schedule:

- Appointment
- Meeting
- Task
- Event

View:

- Daily
- Weekly
- Monthly
- Yearly
- Next
- Previous
- Lists:
  - *o* Appointments
  - *o* Meetings
  - *o* Tasks
  - *o* Events
- Goto
- Filter
- Other User
- Windows

Admin

- Users
- Groups
- Rooms
- Global Options:
  - *o* Times & Dates
  - *o* Categories
  - *o* Views

Options:

- Times & Dates
- Categories
- Views

Sections 2.2 and Beyond

Sections 2.2 and Beyond

- These sections differ for each project.

Sections 2.2 and Beyond

- These sections differ for each project.
- For Milestone 2 they’re rough and preliminary.

Sections 2.2 and Beyond

- These sections differ for each project.
- For Milestone 2 they’re rough and preliminary.
    - Calendar example is top-down in style.

Sections 2.2 and Beyond

- These sections differ for each project.
- For Milestone 2 they're rough and preliminary.
  - Calendar example is top-down in style.
  - I.e., a detailed outline has been completed.

2.2 and Beyond, cont'd

- Organizational guidelines:

2.2 and Beyond, cont’d

- Organizational guidelines:
  - Generally, organize per functional hierarchy.

2.2 and Beyond, cont’d

- Organizational guidelines:
  - Generally, organize per functional hierarchy.
  - Refine organization with stylistic guidelines, to make document more readable.

2.2 and Beyond, cont’d

- Stylistic guidelines include:

2.2 and Beyond, cont’d

- Stylistic guidelines include:
  - Start with common activity "reader warm up".

2.2 and Beyond, cont’d

- Stylistic guidelines include:
  - Start with common activity "reader warm up".
  - Simple scenarios first, details later.

2.2 and Beyond, cont'd

- Stylistic guidelines include:
  - Start with common activity "reader warm up".
  - Simple scenarios first, details later.
  - Separate scenarios for different user groups.

2.2 and Beyond, cont’d

- Stylistic guidelines include:
  - Start with common activity "reader warm up".
  - Simple scenarios first, details later.
  - Separate scenarios for different user groups.
  - Leave mundane details to later, e.g., File, Edit.

2.2 and Beyond, cont’d

- Stylistic guidelines include:
  - Start with common activity "reader warm up".
  - Simple scenarios first, details later.
  - Separate scenarios for different user groups.
  - Leave mundane details to later, e.g., File, Edit.
  - Leave details of error handling until later.

2.2 and Beyond, cont’d

- Scenario details:

2.2 and Beyond, cont’d

- Scenario details:
  - Typically shows user selecting an operation.

2.2 and Beyond, cont’d

- Scenario details:
  - Typically shows user selecting an operation.
  - Start with "... the user selects ...".

2.2 and Beyond, cont’d

- Scenario details:
  - Typically shows user selecting an operation.
  - Start with "... the user selects ...".
  - Show resulting screen shot.

2.2 and Beyond, cont’d

- Scenario details:
  - Typically shows user selecting an operation.
  - Start with "... the user selects ...".
  - Show resulting screen shot.
  - Explain screen contents in follow-on narrative.

2.2 and Beyond, cont’d

- Scenario details:
  - Typically shows user selecting an operation.
  - Start with "... the user selects ...".
  - Show resulting screen shot.
  - Explain screen contents in follow-on narrative.
  - Continue in this style, showing user action and results, with generous explanatory narrative.

Section 2.2: Scheduling Appointment

Section 2.2: Scheduling Appointment

- This Calendar example is a typical rough draft.

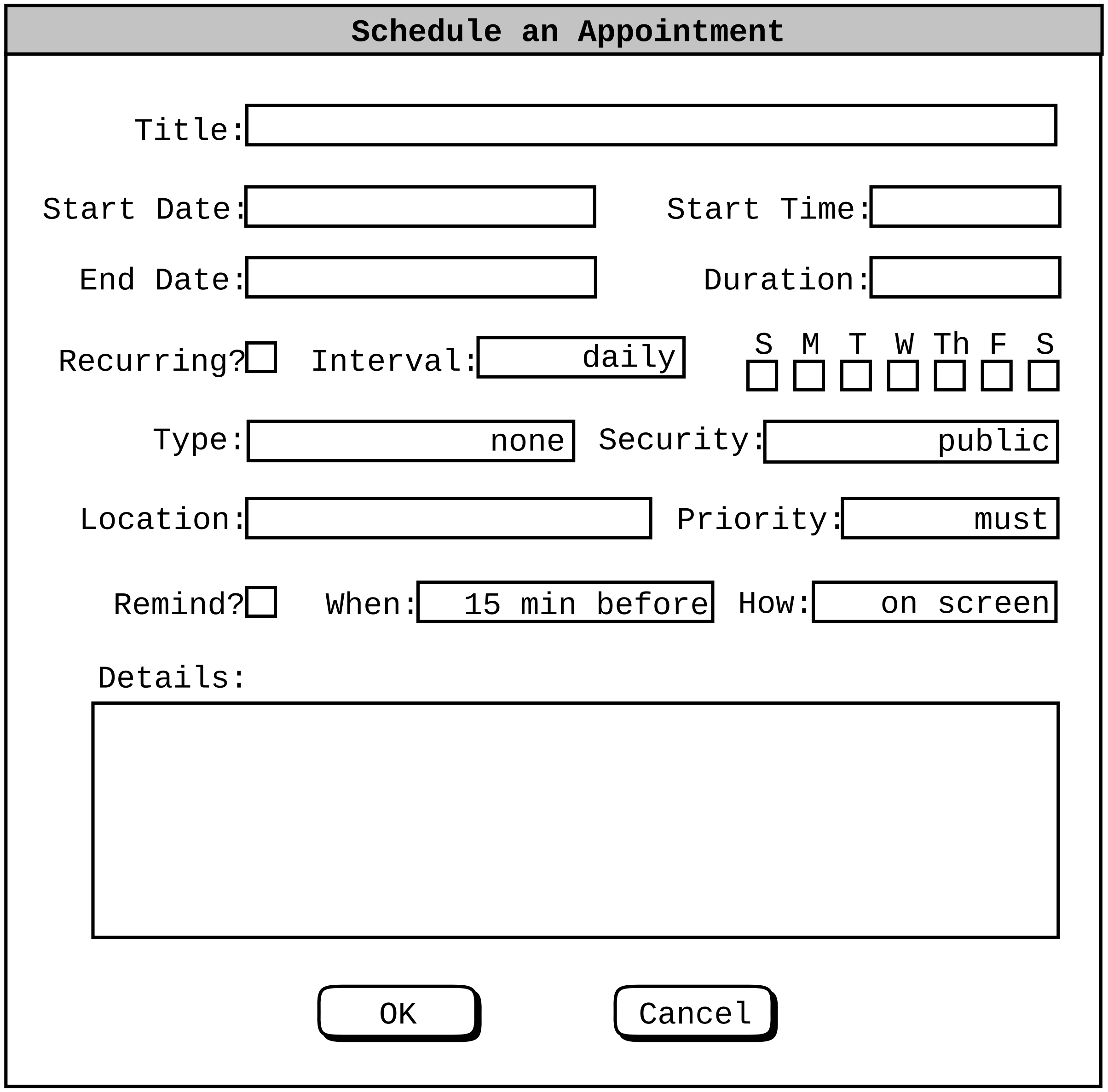
Section 2.2: Scheduling Appointment

- This Calendar example is a typical rough draft.

- Figure 3 shows result of selecting 'Schedule->Appointment'.

Section 2.2: Scheduling Appointment

- This Calendar example is a typical rough draft.
- Figure 3 shows result of selecting ’`Schedule->Appointment`’.
- Explanatory narrative follows.



Schedule an Appointment

Title:			
Start Date:		Start Time:	
End Date:		Duration:	
Recurring? ☐	Interval: daily	S M T W Th F S	☐ ☐ ☐ ☐ ☐ ☐ ☐
Type:	none	Security:	public
Location:		Priority:	must
Remind? ☐	When: 15 min before	How:	on screen

Details:

Figure 3: Appointment Scheduling Dialog

Scheduling Appointment, cont’d

Typical explanatory narrative following screen:

The title field is a one-line string that describes the appointment briefly. The date is the date on which the appointment is to occur. ...

Scheduling Appointment, cont’d

- Figures 4-7 show additional user actions.

Scheduling Appointment, cont’d

- Figures 4-7 show additional user actions.
- Explanatory narrative between each screen shot.

Scheduling Appointment, cont’d

- Figures 4-7 show additional user actions.
- Explanatory narrative between each screen shot.
- It goes like this ...

Scheduling Appointment, cont’d

... user selects `Type:` drop-down ...

Scheduling Appointment, cont’d

... user selects `Type:` drop-down ...

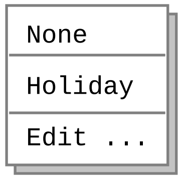


Figure 4: Initial categories menu.

Scheduling Appointment, cont'd

... user selects `Type:` drop-down ...

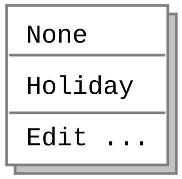


Figure 4: Initial categories menu.

Explanatory narrative ...

Scheduling Appointment, cont’d

... user selects `’Edit ...’`

Scheduling Appointment, cont’d

... user selects `'Edit ...'`

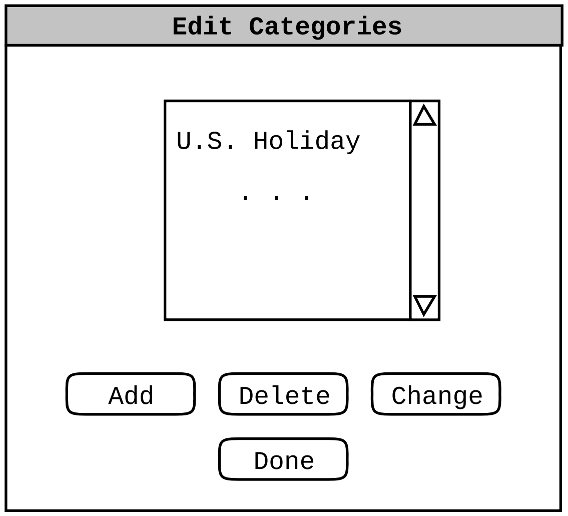


Figure 5: Edit categories dialog.

Scheduling Appointment, cont'd

... user selects `'Edit ...'`

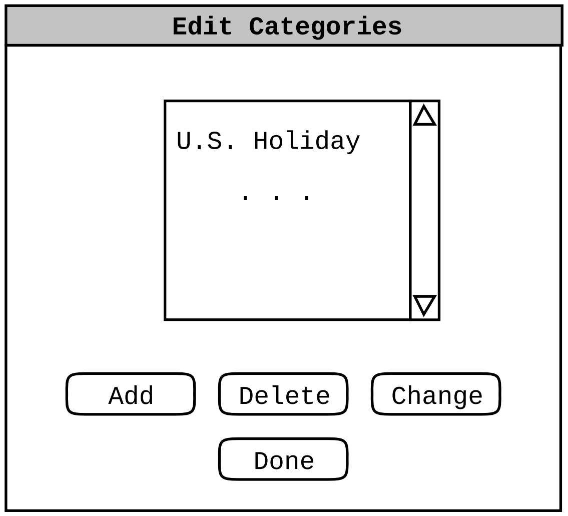


Figure 5: Edit categories dialog.

Explanatory narrative ...

Scheduling Appointment, cont’d

- *Explanatory narrative* will become more refined.

Scheduling Appointment, cont’d

- *Explanatory narrative* will become more refined.
- Eventually, all commands and data formats are covered at least once.

Scheduling Appointment, cont'd

- *Explanatory narrative* will become more refined.
- Eventually, all commands and data formats are covered at least once.
- We'll discuss further in upcoming lectures.

Section 2.3. Browsing

Section 2.3. Browsing

- Editorial remark explains that this and remaining sections are skeletons.

Section 2.3. Browsing

- Editorial remark explains that this and remaining sections are skeletons.

- A number of browsing scenarios are planned.

Section 2.3. Browsing

- Editorial remark explains that this and remaining sections are skeletons.
- A number of browsing scenarios are planned.
- Scenario order generally follows layout of commands in ’`View`’ menu.

2.3 Browsing an Indvidual Calendar

2.3.1 Viewing by Day, Week, Month, and Year

2.3.2 Viewing Lists of Scheduled Items

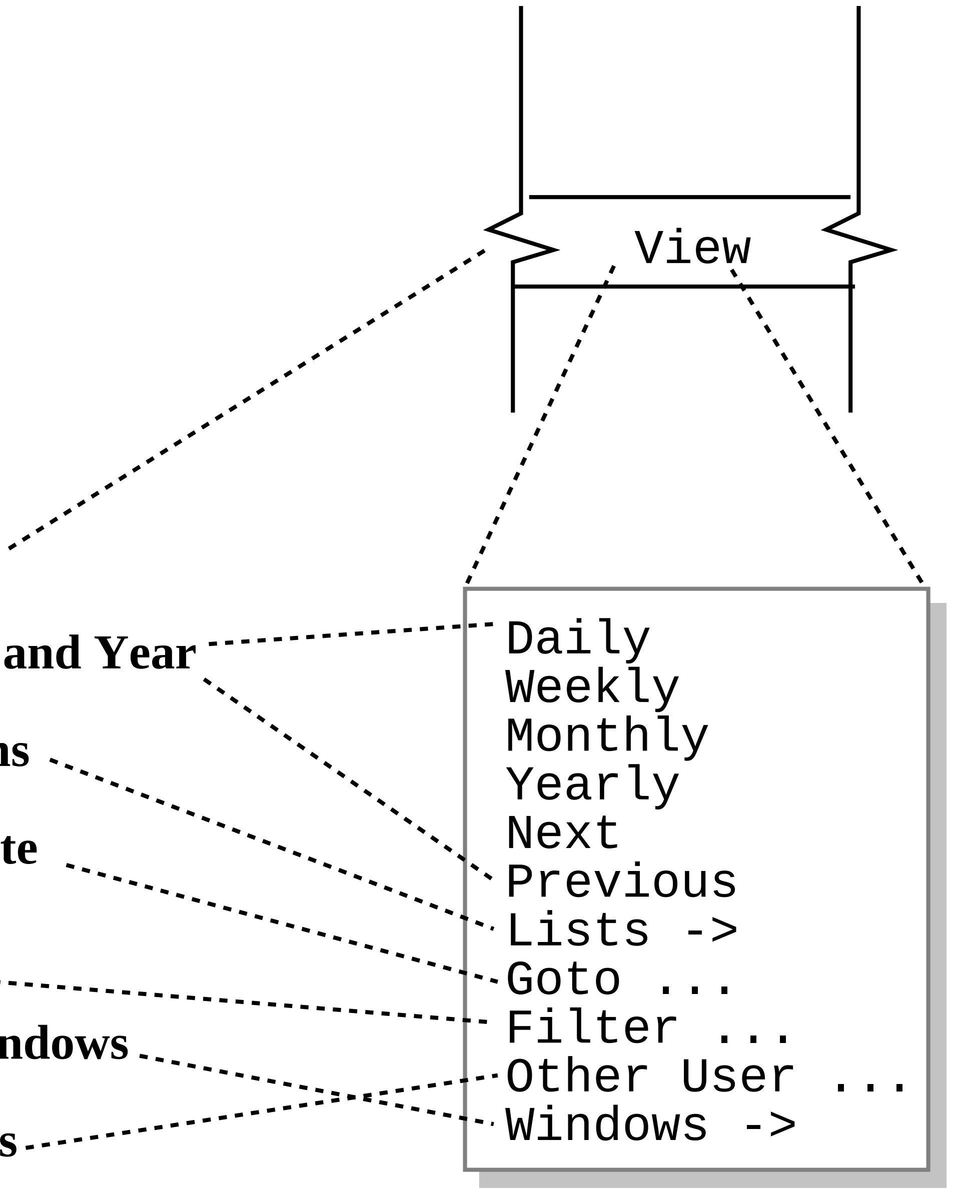
2.3.3 Viewing a Specific Calendar Date

2.3.4 Filtered Viewing

2.3.5 Viewing Recently Displayed Windows

2.3.6 Viewing Other Users' Calendars

2.3.7 Receiving Reminders



Critique of Section 2.3
Rough Draft Organization

Critique of Section 2.3 Rough Draft Organization

- For consistency, use term "Viewing" instead of "Browsing".

Critique of Section 2.3 Rough Draft Organization

- For consistency, use term "Viewing" instead of "Browsing".
- Section 2.3.1 may get too big.

Critique of Section 2.3
Rough Draft Organization

- For consistency, use term "Viewing" instead of "Browsing".
- Section 2.3.1 may get too big.
- Flip order of 2.3.5 and 2.3.6 to be consistent with functional hierarchy.

Critique of Section 2.3 Rough Draft Organization

- For consistency, use term "Viewing" instead of "Browsing".
- Section 2.3.1 may get too big.
- Flip order of 2.3.5 and 2.3.6 to be consistent with functional hierarchy.
- Minor details at this point, but worth noting.

Section 2.4. More Scheduling

Section 2.4. More Scheduling

- These scenarios cover remaining commands in ’`Schedule`’ menu.

Section 2.4. More Scheduling

- These scenarios cover remaining commands in ’`Schedule`’ menu.

- Stylistically, the "simple-to-more-detailed" guide-line is being used here.

Section 2.4. More Scheduling

- These scenarios cover remaining commands in ’`Schedule`’ menu.

- Stylistically, the "simple-to-more-detailed" guide-line is being used here.

  - *o* I.e., start with simple scenario on basic schedul-ing (Section 2.2).

Section 2.4. More Scheduling

- These scenarios cover remaining commands in ’`Schedule`’ menu.

- Stylistically, the "simple-to-more-detailed" guide-line is being used here.

  - *o* I.e., start with simple scenario on basic schedul-ing (Section 2.2).

  - *o* Cover remaining details subsequently.

Section 2.5. Scheduling Group Meetings

Section 2.5. Scheduling Group Meetings

- This scenario covers scheduling from a group leader’s perspective.

Section 2.5. Scheduling Group Meetings

- This scenario covers scheduling from a group leader’s perspective.

- Stylistically, the "user-category" guideline is being used here.

Section 2.5. Scheduling Group Meetings

- This scenario covers scheduling from a group leader’s perspective.

- Stylistically, the "user-category" guideline is being used here.

  - I.e., start with scheduling scenario for most common user category (registered user).

Section 2.5. Scheduling Group Meetings

- This scenario covers scheduling from a group leader’s perspective.

- Stylistically, the "user-category" guideline is being used here.

  - I.e., start with scheduling scenario for most common user category (registered user).

  - Present subsequent advanced scenarios.

Section 2.6. Admin Functions

Section 2.6. Admin Functions

- Scenarios for ’`Admin`’ menu commands.

Section 2.6. Admin Functions

- Scenarios for ’`Admin`’ menu commands.
- Stylistically, things come together naturally here.

Section 2.6. Admin Functions

- Scenarios for '`Admin`' menu commands.
- Stylistically, things come together naturally here.
  - Follow the functional command hierarchy.

Section 2.6. Admin Functions

- Scenarios for ’`Admin`’ menu commands.
- Stylistically, things come together naturally here.
  - *o* Follow the functional command hierarchy.
  - *o* Commands for different user category (admin).

Section 2.6. Admin Functions

- Scenarios for ’`Admin`’ menu commands.
- Stylistically, things come together naturally here.
  - *o* Follow the functional command hierarchy.
  - *o* Commands for different user category (admin).
  - *o* Somewhat mundane operations towards end.

Sections 2.7 and 2.8. Options, File, Edit

Sections 2.7 and 2.8. Options, File, Edit

- Again, we’re following the "mundane details towards end" guideline.

Sections 2.7 and 2.8. Options, File, Edit

- Again, we’re following the "mundane details towards end" guideline.

- These details are important, but not what the Calendar Tool is mainly about.

Sections 2.7 and 2.8. Options, File, Edit

- Again, we’re following the "mundane details towards end" guideline.
- These details are important, but not what the Calendar Tool is mainly about.
- The point is, we try to keep the reader engaged without compromising overall organization.

Sections 2.7 and 2.8. Options, File, Edit

- Again, we’re following the "mundane details towards end" guideline.

- These details are important, but not what the Calendar Tool is mainly about.

- The point is, we try to keep the reader engaged without compromising overall organization.

- Use your own good judgment for your projects.

Where Things Stand with Milestone 2

Where Things Stand with Milestone 2

- A very rough draft.

Where Things Stand with Milestone 2

- A very rough draft.
- Focus on fundamental functionality.

Where Things Stand with Milestone 2

- A very rough draft.
- Focus on fundamental functionality.
- Error conditions not yet considered.

Where Things Stand with Milestone 2

- A very rough draft.
- Focus on fundamental functionality.
- Error conditions not yet considered.
- Much work yet to do.

Three Bits of General Information ...

Bi-Weekly Reports

Bi-Weekly Reports

- See template at 307 Handouts page.

Bi-Weekly Reports

- See template at 307 Handouts page.
- Send as plain text email.

Bi-Weekly Reports

- See template at 307 Handouts page.
- Send as plain text email.
- Mail message subject: "307 Report"

Piazza

Piazza

- Sean has set up Piazza for 307.

Piazza

- Sean has set up Piazza for 307.
- Has anyone received an invite?

Drawing Editors

Drawing Editors

- General-purpose drawing tools work fine.

Drawing Editors

- General-purpose drawing tools work fine.
- **Pencil** looks good for 307

Drawing Editors

- General-purpose drawing tools work fine.
- **Pencil** looks good for 307
- **Moqups** and **Balsamiq** are also popular

Drawing Editors

- General-purpose drawing tools work fine.
- **Pencil** looks good for 307
- **Moqups** and **Balsamiq** are also popular
- **Visio** is good for Windows

Drawing Editors

- General-purpose drawing tools work fine.
- **Pencil** looks good for 307
- **Moqups** and **Balsamiq** are also popular
- **Visio** is good for Windows
- You can use JavaFX **Scene Builder**, but ***don’t*** write any Java code yet.

Now back to an earlier week 2 lecture topic ...