CSC 308

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. Projects descriptions.

I. Materials for weeks 1 and 2:

A. Syllabus.

B. Projects descriptions.

C. Milestone 1 writeup.

I. Materials for weeks 1 and 2:

A. Syllabus.

B. Projects descriptions.

C. Milestone 1 writeup.

D. Specification document outline.

I. Materials for weeks 1 and 2:

A. Syllabus.

B. Projects descriptions.

C. Milestone 1 writeup.

D. Specification document outline.

E. SVN basics.

I. Materials for weeks 1 and 2:

A. Syllabus.

B. Projects descriptions.

C. Milestone 1 writeup.

D. Specification document outline.

E. SVN basics.

F. Standard operating procedures, Volume 1.

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. Choice of project teams and projects.

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. Choice of project teams and projects.
   b. Prep for initial customer interviews.

Scheduling, cont'd

B. Second day (Wed):

Scheduling, cont'd

B. Second day (Wed):

   1. Customer interviews, both lecture & lab.

Scheduling, cont'd

B. Second day (Wed):

   1. Customer interviews, both lecture & lab.
   2. No normal lecture.

Scheduling, cont'd

B. Second day (Wed):

1. Customer interviews, both lecture & lab.
2. No normal lecture.
3. Schedule TBA, on Mon.

Scheduling, cont'd

C. Third day (Fri):

Scheduling, cont'd

C. Third day (Fri):

   1. Normal lecture.

Scheduling, cont'd

C. Third day (Fri):

1. Normal lecture.
2. Lab intro to project repository and SVN.

Scheduling, cont'd

D. Fourth day (Mon, Week 2):

Scheduling, cont'd

D. Fourth day (Mon, Week 2):

    1. Second round of customer interviews.

Scheduling, cont'd

D. Fourth day (Mon, Week 2):

   1. Second round of customer interviews.

   2. As with preceding Wed, no normal lecture.

Scheduling, cont'd

D. Fourth day (Mon, Week 2):

   1. Second round of customer interviews.
   2. As with preceding Wed, no normal lecture.
   3. Precise schedule TBA.

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:

WF 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

Projects

Projects

- Theme is classroom software.

Projects

- Theme is classroom software.
- Replacements for PolyLearn products.

Projects

- Theme is classroom software.
- Replacements for PolyLearn products.
- The projects are:

Projects

- Theme is classroom software.
- Replacements for PolyLearn products.
- The projects are:
  1. *Grader*

Projects

- Theme is classroom software.
- Replacements for PolyLearn products.
- The projects are:
  1. *Grader*
  2. *Test Tool*

Projects

- Theme is classroom software.
- Replacements for PolyLearn products.
- The projects are:
  1. *Grader*
  2. *Test Tool*
  3. *Electric Classroom*

Projects

- Theme is classroom software.
- Replacements for PolyLearn products.
- The projects are:
  1. *Grader*
  2. *Test Tool*
  3. *Electric Classroom*
  4. *Class Scheduler*

Projects

- Theme is classroom software.
- Replacements for PolyLearn products.
- The projects are:
  1. *Grader*
  2. *Test Tool*
  3. *Electric Classroom*
  4. *Class Scheduler*
  5. *CS Tutor*

Milestone 1

Milestone 1

- **Due:** 2nd week, *11:59:59PM Wednesday*

Milestone 1

- **Due:** 2nd week, *11:59:59PM Wednesday*
- **Tasks:**

Milestone 1

- **Due:** 2nd week, *11:59:59PM Wednesday*
- **Tasks:**
  1. Organize team

Milestone 1

- **Due:** 2nd week, *11:59:59PM Wednesday*
- **Tasks:**
  1. Organize team
  2. Brainstorm about tool features

Milestone 1

- **Due:** 2nd week, *11:59:59PM Wednesday*
- **Tasks:**
  1. Organize team
  2. Brainstorm about tool features
  3. Look for related tools

Milestone 1

- **Due:** 2nd week, *11:59:59PM Wednesday*
- **Tasks:**
  1. Organize team
  2. Brainstorm about tool features
  3. Look for related tools
  4. Prepare week 2 interview questions

Milestone 1

- **Due:** 2nd week, *11:59:59PM Wednesday*
- **Tasks:**
  1. Organize team
  2. Brainstorm about tool features
  3. Look for related tools
  4. Prepare week 2 interview questions
  5. Do rough draft Section 1 of requirements

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 308, 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 308, you are primarily analysts, secondarily domain experts and end users.

E. Program design and imple'n happens in 309.

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.

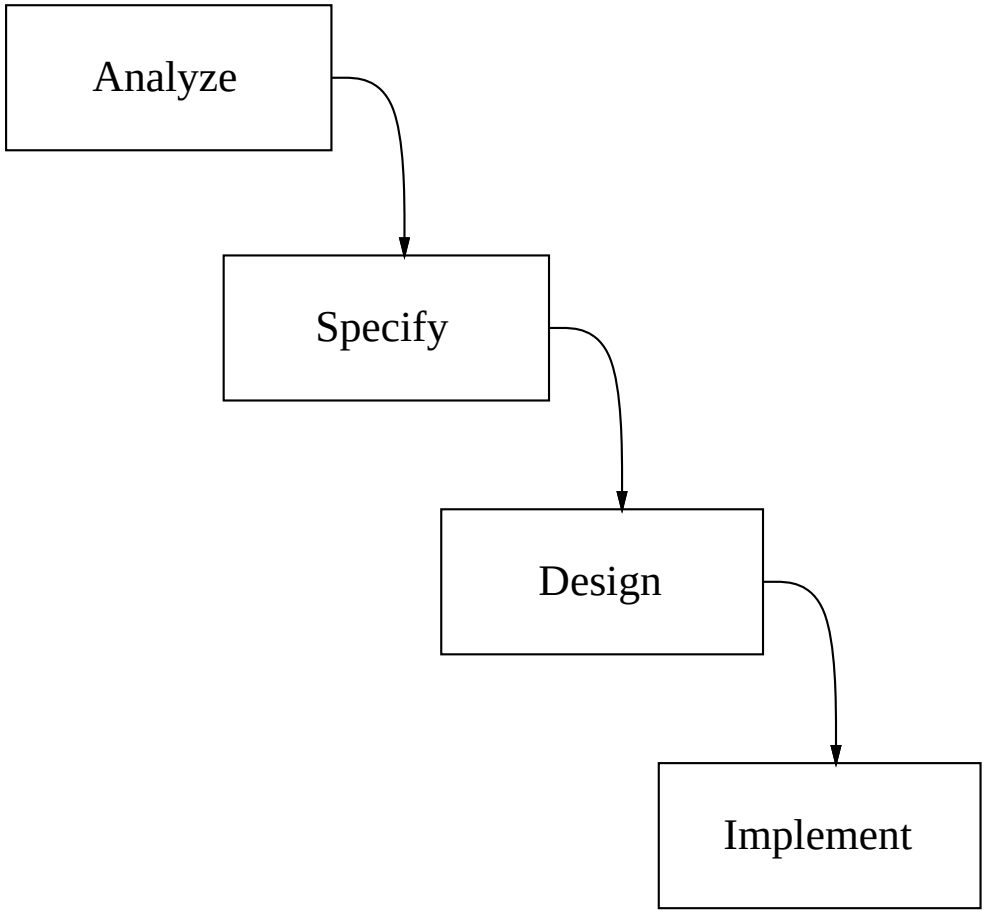


Figure 1: Major phases of SE process.

Process, cont'd

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

Process, cont'd

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

1. Acquire and organize functional require-ments 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.

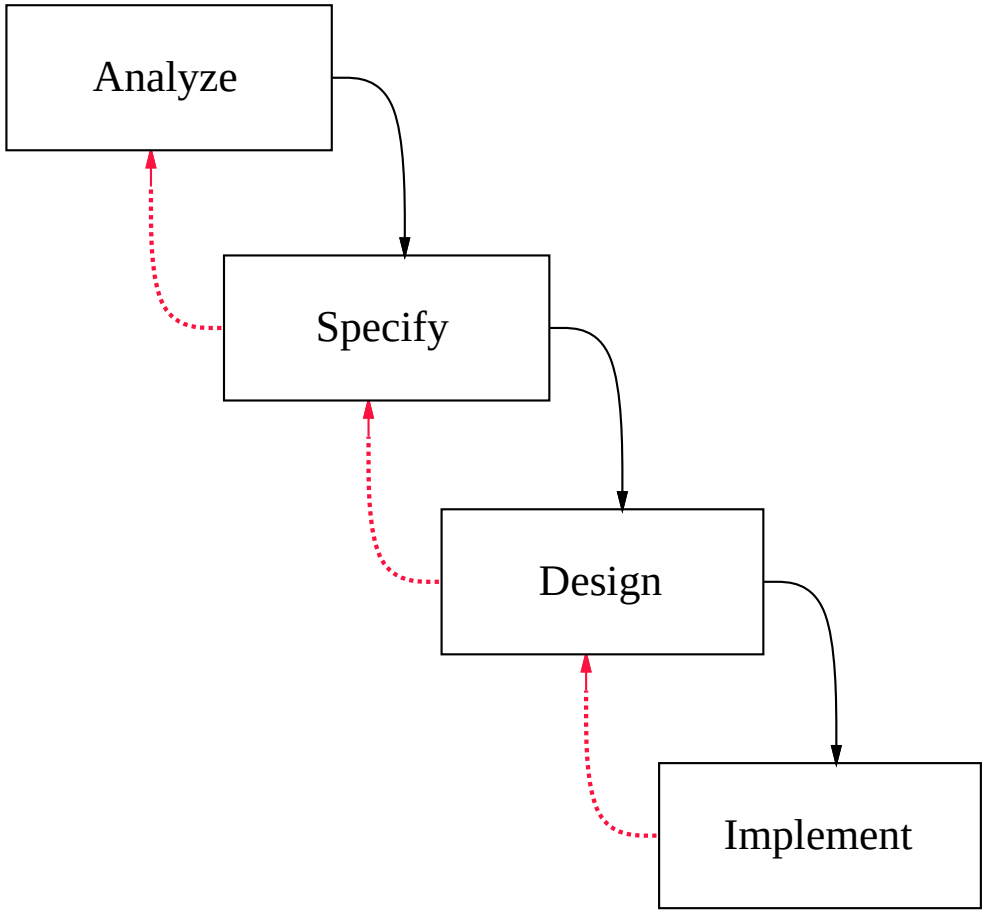


Figure 1: Updated SE process.

Process, cont'd

3. In the 308/309 process:

Process, cont'd

3. In the 308/309 process:

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

Process, cont'd

3. In the 308/309 process:
   a. Much feedback between **Analyze** & **Specify**
   b. Much feedback between **Design** & **Imple**

Process, cont'd

3. In the 308/309 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***

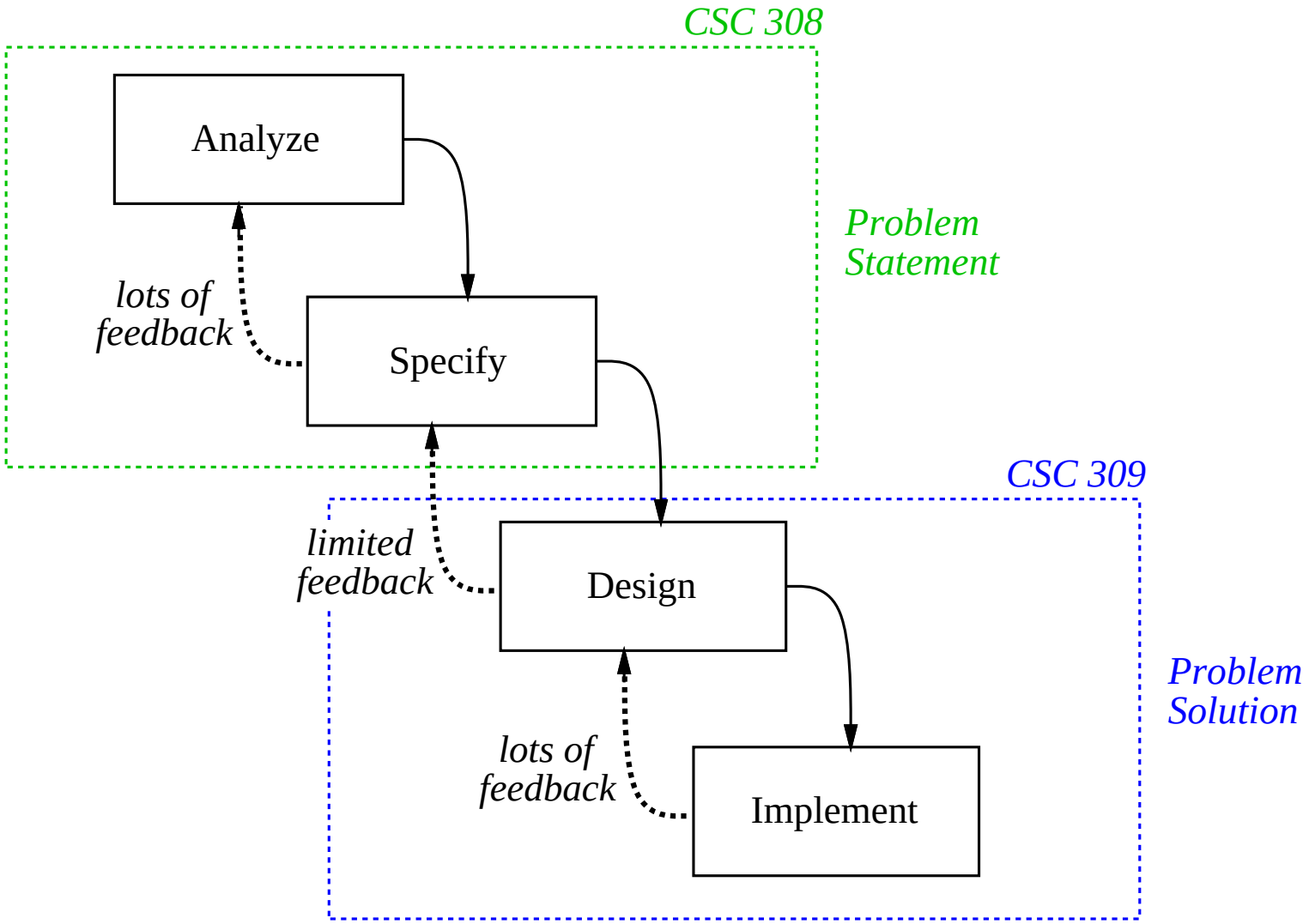
Process, cont'd

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 intervals.

Pervasive steps, cont'd

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

1. Project meetings are scheduled at regular intervals.

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 suitable for everyone involved.

1. Requirements spec document.

Pervasive steps, cont'd

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

1. Requirements spec document.

2. Maintenance documentation.

Pervasive steps, cont'd

H. The **Document** step produces documents suitable 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. 308/309 process considered *traditional*.

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

A. 308/309 process considered *traditional*.

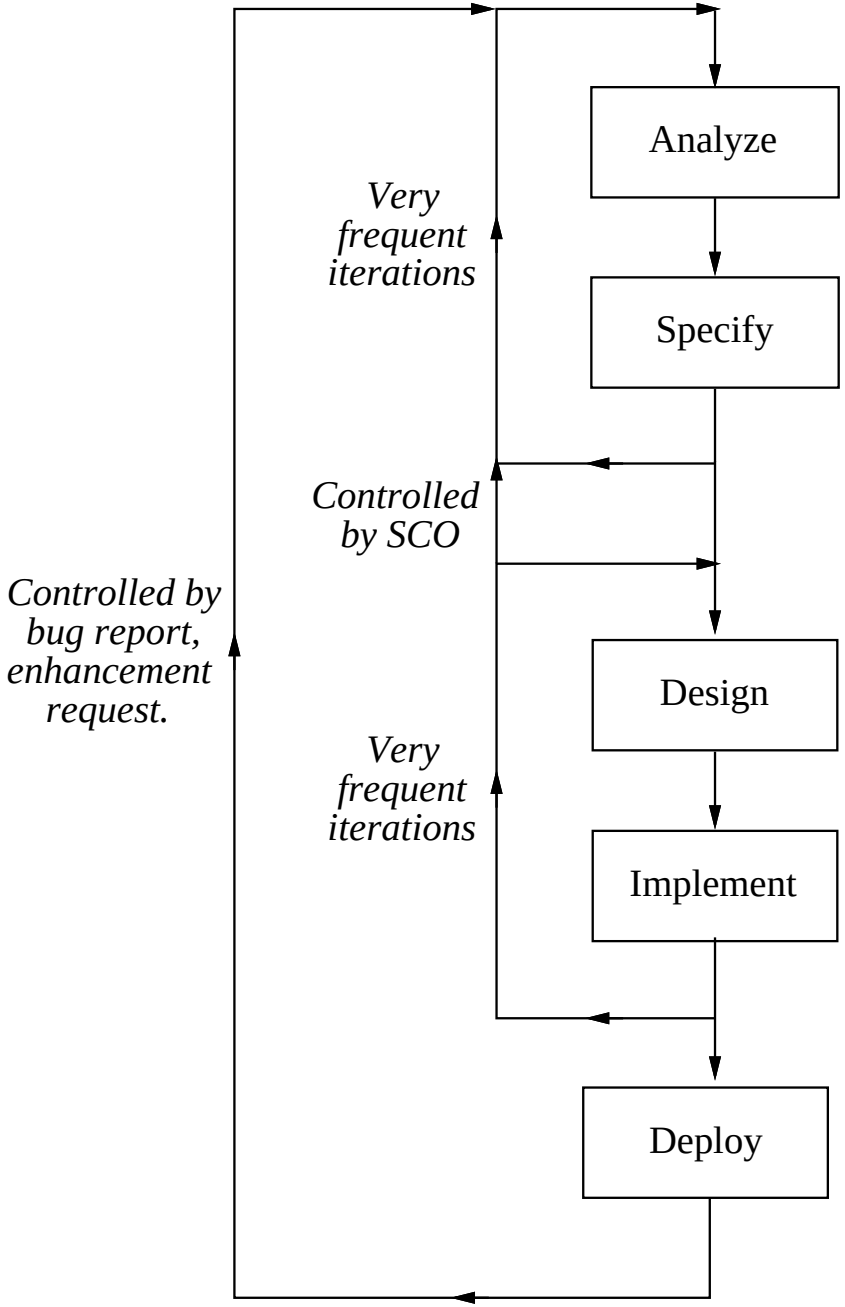
B. Particularly the production of a substantial requirements document.

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

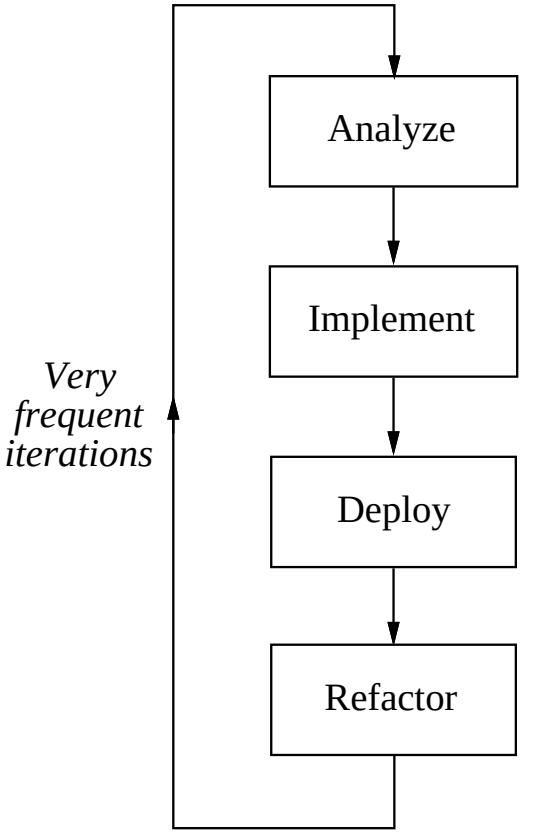
A. 308/309 process considered *traditional*.

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 deveopment, or *extreme programming*:

Traditional versus agile, cont'd

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

1. Customers and implementors work very closely together.

Traditional versus agile, cont'd

D. In agile deveopment, 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.**

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. Organizations learn that hastily-acquired systems can cause problems.

E. Companies find insubstantial markets for their software products.

F. Nearly universal agreement that thorough requirements analysis is important.

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

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.

1. Formal spec language is used.

2. Primary audience is system design/implementation team.

3. Final version is a *very* formal.

XIII. Details of user consultations

A. Critically important to involve end-users in requirements process.

B. Success is far more likely.

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

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**

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**

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.

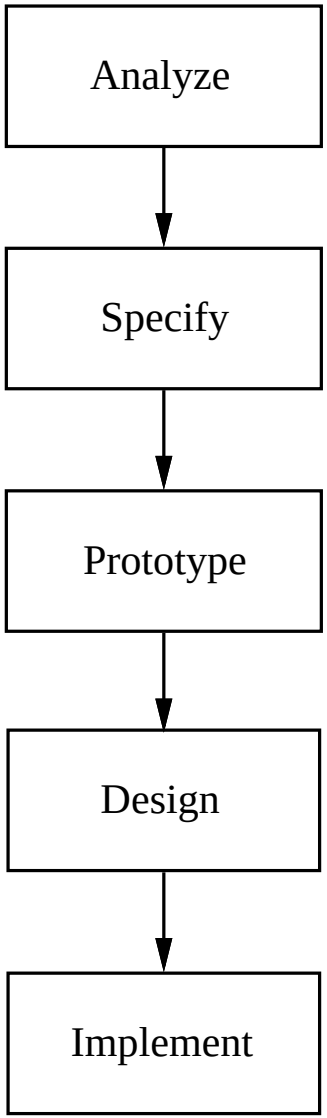
2. Screens, commands, data formats, and all other user-visible aspects of operation.

XVII. **Rapid system prototyping**

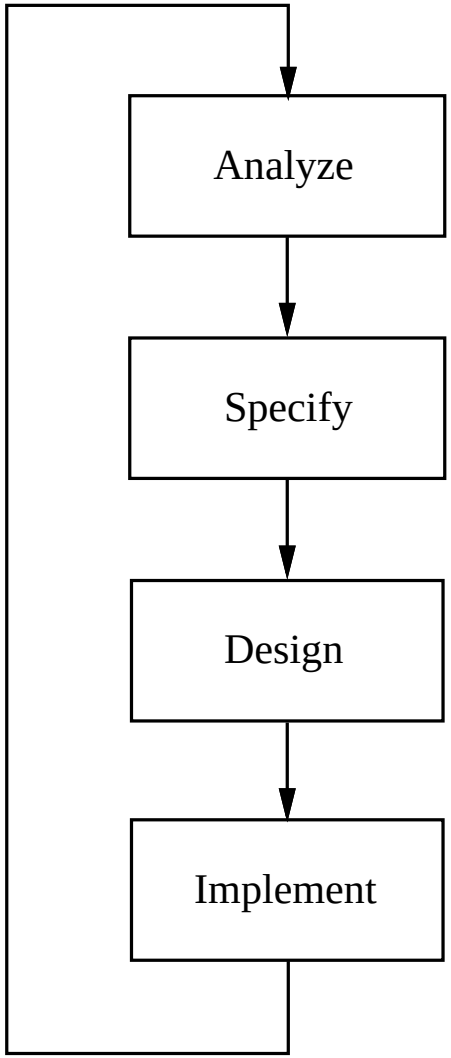
A. Can help capture user requirements.

B. Version 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 308/309, we'll do both styles

- We'll do a bit of GUI prototyping in 308, as in Figure 2b.
- Overall, the 309 product can be considered an operational prototype, as in Figure 2a.

XVIII. **Establishing genuine user needs**

A. Quite critical.

B. Plenty of software has been built without sufficiently demonstrated need.

C. Forthright analyst should be prepared to say to customers "You don’t need new software"

D. Marketing analysts must be prepared to recognize insubstantial market.

XIX. **Other important aspects**

A. Identification of personnel.

B. Overview of current and proposed operations.

C. Analysis of relevant existing systems.

D. Impact analysis.

XX. Examples of requirements specification

XX. Examples of requirements specification

A. Concrete example similar in size and scope to your 308 projects.

XX. Examples of requirements specification

A. Concrete example similar in size and scope to your 308 projects.

B. Example presented in phases corresponding to milestones.

XX. Examples of requirements specification

A. Concrete example similar in size and scope to your 308 projects.

B. Example presented in phases corresponding to milestones.

C. First example covers Milestone 1.

XX. Examples of requirements specification

A. Concrete example similar in size and scope to your 308 projects.

B. Example presented in phases corresponding to milestones.

C. First example covers Milestone 1.

D. We'll go over in detail.

Milestone 1 Writeup

Milestone 1 Writeup

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

Milestone 1 Writeup

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

Milestone 1 Writeup

- Due second week, check in by 11:59PM Wed
- Tasks:
  a. Team duties

Milestone 1 Writeup

- Due second week, check in by 11:59PM Wed
- Tasks:
  a. Team duties
  b. Brainstorming

Milestone 1 Writeup

- Due second week, check in by 11:59PM Wed
- Tasks:
  a. Team duties
  b. Brainstorming
  c. Tools search

Milestone 1 Writeup

- Due second week, check in by 11:59PM Wed
- Tasks:
  a. Team duties
  b. Brainstorming
  c. Tools search
  d. Questions for week 2 customer interview

Milestone 1 Writeup

- Due second week, check in by 11:59PM Wed
- Tasks:
  a. Team duties
  b. Brainstorming
  c. Tools search
  d. Questions for week 2 customer interview
  e. 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 providing 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

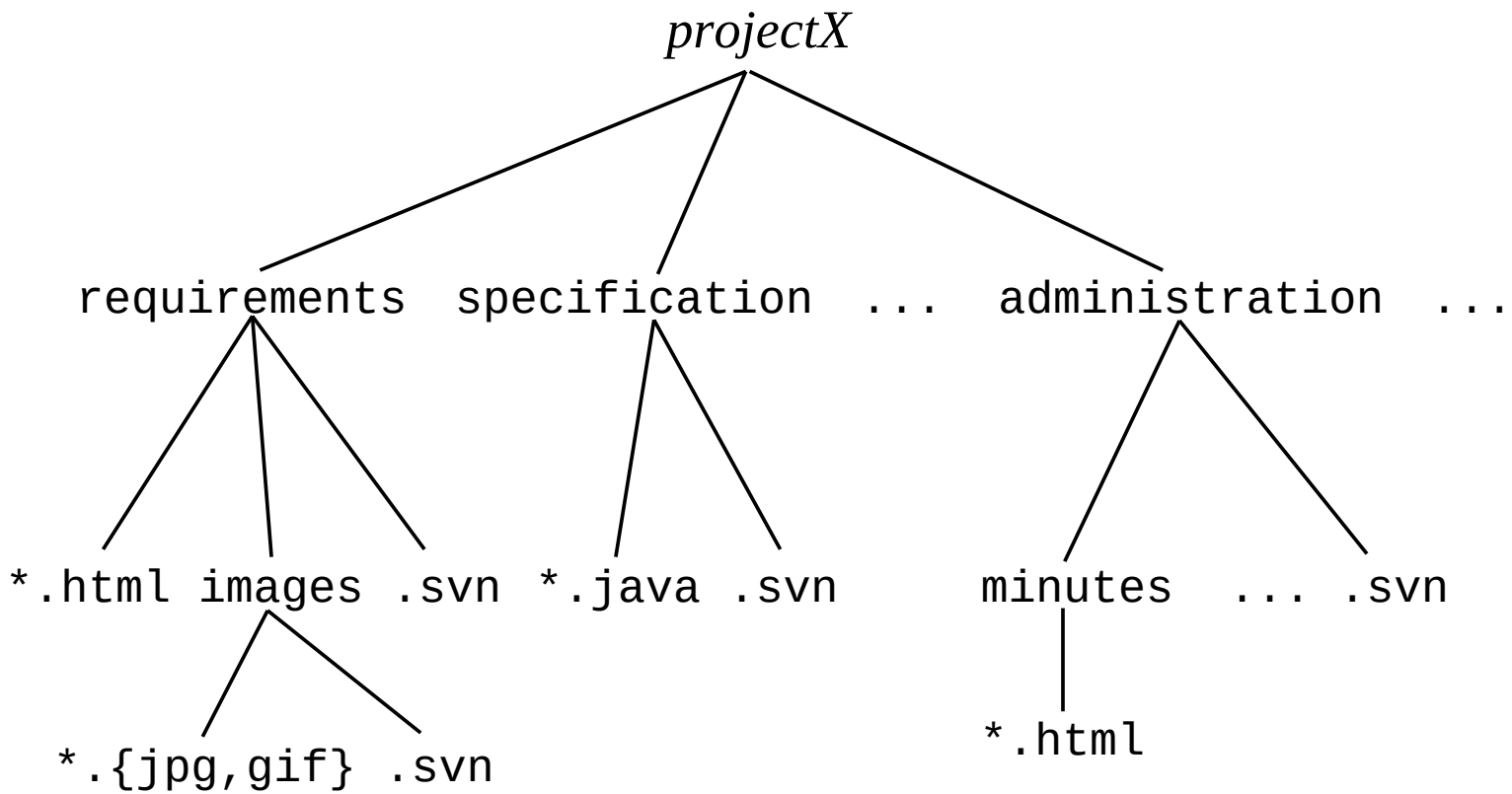
- Other software with similar functionality.
- Consider:
  - What is good about them.
  - What is bad.

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:
  - removing unnecessary files
  - checking file status

SVN Basics, cont'd

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