CS3205:

Task Analysis and Techniques

CS3205: Task Analysis and Techniques

Readings (same as before):

1) ID-Book Chapter "Establishing Requirements," Ch. 10 (Ch. 9 in course eBook)

2) Chapter 2 from Task-Centered User Interface Design, http://hcibib.org/tcuid/chap-2.html

Task Analysis

- Same as requirements analysis?
  - Focus on users, not on the proposed system
  - “Earlier” than “traditional” req. analysis
  - But lots in common…

Things to Note in Ch. 2, TCUID

- Examples of “complete jobs” idea
- Final hypertopic:
  Defining tasks vs. traditional requirements and specification
  - Real or representative vs. abstract.
  - Complete vs. partial (from user point of view)

Functional Requirements

- See Figure 10.1 -- a classic way of describing functional requirements

Goals vs. Tasks vs. Actions

- Goal
  - end result to be achieved
- Task
  - structured set of related activities undertaken in a sequence
- Action
  - one step or action performed (part of a task)

Do Tasks Have Important Characteristics?

- Characteristics:
  - Vary by occasion?
  - Carried out frequently, once, etc.?
  - Time-critical?
  - Users work alone or with others?
  - Switching between tasks?
  - Etc.
- Are there multiple ways to do tasks?
  - Multiple sequences of tasks
- Multiple tasks that achieve a goal?

“Doing” Task Analysis

- Different levels of granularity
  - Recognize this.
  - Use methods that can deal with this and capture various levels
- A organizational level: interactions between multiple people
  - Workflow analysis:
    - Describing document/information flow between people
    - Sequence, dependencies

“Doing” Task Analysis (2)

- Specific techniques? Sure! Lots!
- Do you want to:
  - Describe the steps to complete a task? (What is to be done)
  - Capture what knowledge or information people need to complete a task? (More of how a user does a task.)

Describing Tasks and Requirements

- HTA form of Task Analysis
- Stories (from Extreme Programming)
- Scenarios
  - an informal narrative story, simple, 'natural', personal, not generalizable
- Use cases
  - assume interaction with a system
  - assume detailed understanding of the interaction
- Essential Use Cases
  - abstract away from the details
  - does not have the same assumptions as use cases

Principles of Good Techniques

- User-centered, not developer centered
  - Say what the user wants to do, not how
  - Avoid UI assumptions
  - Be as specific as possible
    - OK even desirable to talk in examples (see TCUID)
    - E.g. "Bob logs in and searches for backup software for Windows 8 under $50"
  - Start by focusing on tasks that meet user end goals ("complete jobs")
    - Talk about "searching for items" first, rather than how to enter search criteria
    - Interaction between activities/tasks are often problems
  - Link users with tasks
    - Use roles or personas

Task Analysis

- Task descriptions (stories, scenarios, use cases) are often used to envision new systems or devices
- *Task analysis* is used mainly to investigate an existing situation
  - An existing system. Or a process.
  - How does a user currently do a job?
  - Comes from observing and talking to users
- <u>May</u> include tasks, objects outside the computer system
  - E.g. get report, find names, search in system, mark up report

- Task analysis methods <u>should be</u> user-centered

- Many techniques, a well-known on in HCI is Hierarchical Task Analysis (HTA)
  - But in CS3205, others first….

Scenarios

- Is this a formal term or just an informal one?
  - Probably informal
  - In fact, think about how what book says about *scenarios* is like or unlike XP’s *stories*
- By one definition, a scenario is:
  - an informal narrative story
  - simple, ‘natural’, personal
  - not generalizable
- Some use term *Task Scenario*
  - Narrative description of a specific thing done with a current system
  - Like a concrete use-case. (Real, representative)
- See example on next slide and examples in book

Scenario for shared calendar

"The user types in all the names of the meeting participants together with some constraints such as the length of the meeting, roughly when the meeting needs to take place, and possibly where it needs to take place. The system then checks against the individuals' calendars and the central departmental calendar and presents the user with a series of dates on which everyone is free all at the same time. Then the meeting could be confirmed and written into people's calendars. Some people, though, will want to be asked before the calendar entry is made. Perhaps the system could email them automatically and ask that it be confirmed before it is written in."

XP Stories

- Read this one page!
  http://www.extremeprogramming.org/rules/userstories.html
- Short description of system behavior
  - From user point of view
  - Written on <u>one</u> index card!
  - Written by developer and customer together
- Note that what the page says:
  - Level of detail: enough to estimate time to implement (in terms of weeks)
  - "Promises for Conversation" with user
  - Focus on user needs
    - Not on algorithms, UI, database details, technology, etc.

Examples of XP Stories

- When a transaction causes a customer’s account to go into overdraft, transfer money from the overdraft protection account, if any.
- Produce a statement for each account, showing transaction date, number, payee, and amount. A sample statement is attached – make the report look something like the sample.
- When the GPS has contact with two or fewer satellites for more than 60 seconds, display the message “Poor satellite contact” and wait for confirmation by the user. If contact improves before confirmation, clear the message automatically.

More XP Stories (Shorter Ones!)

- Students can purchase monthly parking passes online.
- Parking passes can be paid via credit cards.
- Parking passes can be paid via PayPal.
- Professors can input student marks.
- Students can obtain their current seminar schedule.
- Students can order official transcripts.
- Students can only enroll in seminars for which they have prerequisites.
- Transcripts will be available online via a standard browser.

More Recent Ideas on User Stories

- Read about User Stories in Agile methods here: http://www.mountaingoatsoftware.com/topics/user-stories
- Follow this template:

  As a <user-type>, I want <some goal>
  so that <some reason>.
- Written on index-cards or sticky notes
- Arrange on walls or tables
- Discussed – very important!

Pros and Cons for Stories?

- Pros
  - Leads to more discussion with customer
  - Understandable by a user
- Cons
  - Maybe not enough detail

Pros and Cons for Stories?

- Pros?
  -
- Cons?
  -

Pros and Cons for Stories?

- Pros?
  - Succint
  - Customer tells you what they want, not what you think they want
- Cons?
  - Need more detail

Pros and Cons for Stories?

- Pros?
  - Simple
  - Able to judge priority
  - Understandable by users, from a user POV
  - Doesn’t commit to design too early
  - Support categories, organization
- Cons?
  - Will need detail. Where? When?
  - Could be vague

Pros and Cons for Stories?

- Note: the following list comes from class discussion
- Pros:
  - Users involved.
  - Fast, easy – low overhead.
  - supports planning, iterative development, measuring progress
- Cons:
  - May not be technically enough to insure it comes out the way the user really wants
  - Do they show related behaviors? interactions?
  - Need to talk to many/several stakeholders
  - What if lots of index cards?!

Adding Details to Stories

- See discussion at Mountain Goat site
- High-level stories are called *epics.* Break them into smaller stories.

  As a user, I can backup my drive.

  ➔

  As a user, I can select which folders not to
  backup so I don't fill up my backup drive.

  (and others)

Adding Details to a Story

- What's the "full story" for this user-story?

  "As a user ordering, I want to cancel an order."

- Not a lot of detail. You might have questions like:
  - Does the user get a full or partial refund? Restocking fee? Store credit?
  - Does user get confirmation? How?
  - If it's a multi-item order, can the user cancel some of the items?

Conditions of Satisfaction

- These are statements of what it means to satisfy the story
- Basis for acceptance tests
- AKA “acceptance criteria” for stories
- Might include examples
- Discussed and negotiated with product owner
- Examples from “cancel order” story:
  - Verify user canceling a credit card order is credited on their account.
  - Verify user canceling a cash order has store-credit recorded.
  - Etc.

Example

- Story: "As a customer, I am required to login before using the site"

- Possible conditions of satisfaction:
  - Customer is logged in only when proper credentials are provided
  - A "remember me" option is available
  - Customer can request a password reminder
  - Customer is locked out after three failed attempts

Use Cases

- A more formal definition than a scenario or story
  - From OO techniques, including UML, OOSE, etc.
- Note terms in text:
  - task scenario
  - concrete use case
  - essential use case
  - use scenario

Use Case Modeling

- Use Case:
  - "A sequence of actions a system performs to yield an observable result of value to a particular actor."
- Actor:
  - Someone or something outside the system that interacts with the system
  - A user of the system in a particular role
  - Part of Use Case Modeling is identifying and describing actors
- *Important:* We want an "external view" of the system

Use Cases

- Each use case has a name
  - e.g. Borrrow Copy of Book
- A family (or set, or class) of scenarios
  - One main scenario for “normal” behavior or situation
  - A sequence of interactions documenting that
  - Also set of different but related scenarios
- Documenting Use Cases
  - (Maybe) A UML Diagram showing all of them
    - Actors are stick-figures; use cases are ovals
  - (Certainly) For each use case define using English
    - A clear textual description (like a stories, a scenarios
    - A set of scenarios in outline form

Example: Actors and Use Cases

- Consider a library system…

- Actors
  - BookBorrower
  - JournalBorrower
  - Browser (person who browses, not software)
  - Librarian
- Use Cases
  - Borrow copy of a book
  - Reserve a book
  - Return copy of book
  - Borrow journal
  - Browse
  - Update Catalog

What Form Does a Use Case Take?

- We can *describe* Use Cases in a variety of ways
- First, text paragraphs
- Describes the Actors who participate with the system
- Describes the sequence of events

Example Text Description

- Borrow copy of a book:

  A Bookborrower presents a copy of a book. The system checks that the s/he is a library member, and that s/he has not checked out too many books. If both checks succeed, then the system records that the member now as this copy of the book. Otherwise it refuses the loan.

What Else Is In a Use Case Description?

- Pre- and Post-conditions
  - Values of variables, system conditions, other use cases etc.
- Normal vs. alternative behavior
  - Can be shown in the text description (somehow)
  - Exceptions vs. acceptable alternatives

Example Template for Use Cases

- *Use case number or id:*
- *Use case title:*
- *Text description (a few sentences)*
- *Actors*
- *Preconditions (if applicable):*
- *Flow of Events:*
- *Basic path:*
- *1.First step*
- *2.Second step*
- *3.etc*
- *Alternative Paths:*
  - *Name and short description (in words) of first alternative path/scenario.*
  - *Name and short description (in words) of 2nd alternative path/scenario.*
  - *etc.*
- *Postconditions (if applicable)*
- *Special conditions (if applicable).*

Path for use case for shared calendar

1. The user chooses the option to arrange a meeting.
2. The system prompts user for the names of attendees.
3. The user types in a list of names.
4. The system checks that the list is valid.
5. The system prompts the user for meeting constraints.
6. The user types in meeting constraints.
7. The system searches the calendars for a date that satisfies the constraints.
8. The system displays a list of potential dates.
9. The user chooses one of the dates.
10. The system writes the meeting into the calendar.
11. The system emails all the meeting participants informing them of them appointment

Alternative courses for shared calendar

Some alternative courses:

5. If the list of people is invalid,

   5.1 The system displays an error message.

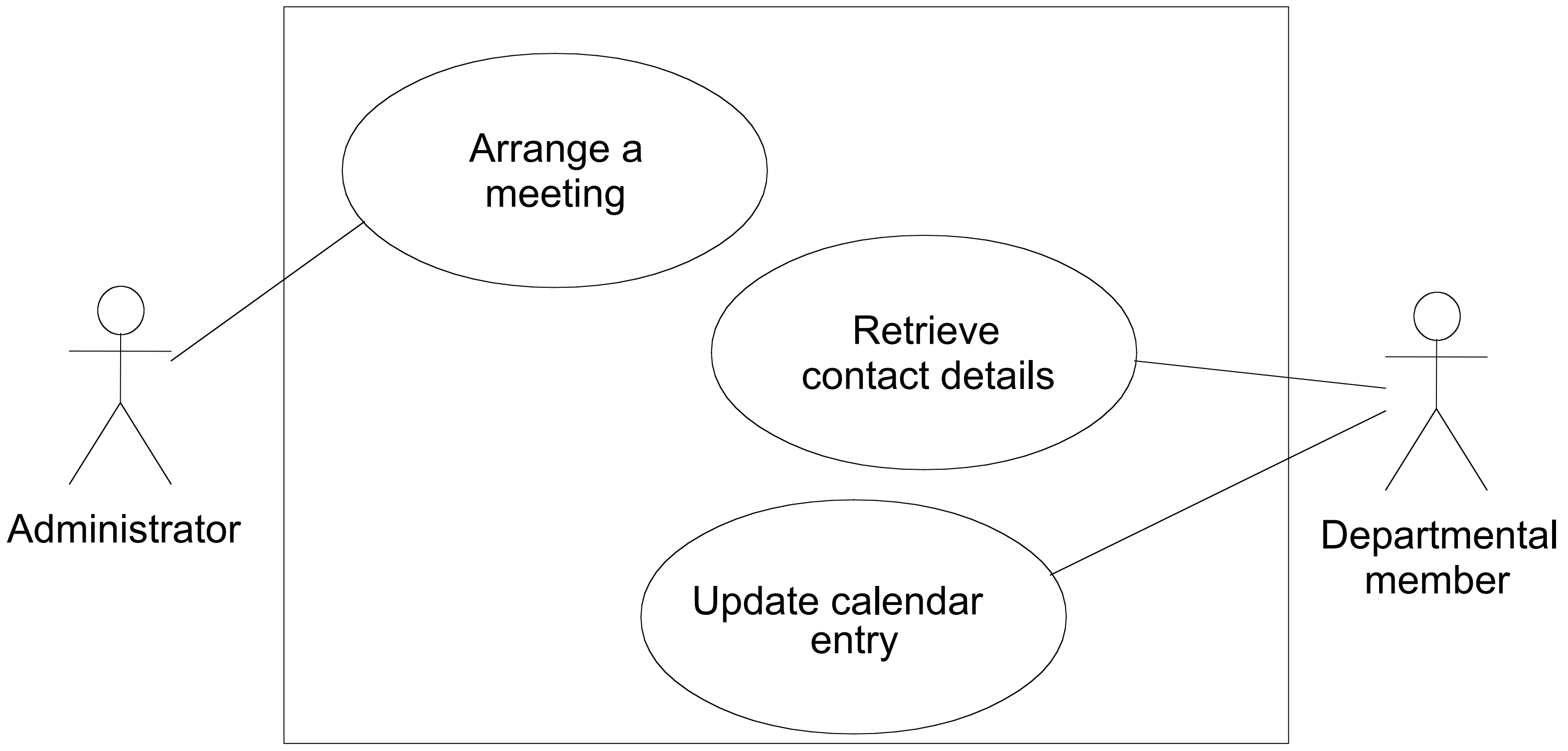
   5.2 The system returns to step 2.

8. If no potential dates are found,

   8.1 The system displays a suitable message.

   8.2 The system returns to step 5.

UML use case diagram for shared calendar



Issues with Use Cases

- Are there problems or issues with use cases?

- Level of detail! We could
  - assume a interaction mode or model
  - specify interactions in terms of UI elements
  - include assumptions about technology
- This is not task-centered, user-centered
  - But, it's not an evil thing. Just do it later in development
- Actors vs. personas
  - Note difference between a general role and a specific imaginary user

Essential Use Cases

- Same idea as use case but…
  - simplified, abstract, generalized
  - captures user tasks
  - without assuming anything about technology or interface implementation
- Shows user intentions, desire
  followed by
  System response

- Essential use cases are at user/system level only

Example essential use case for shared calendar

Name: arrangeMeeting

USER INTENTION	***SYSTEM RESPONSIBILITY***
arrange a meeting	
	request meeting attendees & constraints
identify meeting attendees & constraints	
	search calendars for suitable dates
	suggest potential dates
choose preferred date	
	book meeting

Hierarchical Task Analysis

Tasks and Plans in HTA

- Involves breaking a **task** down into **subtasks**
  - Then sub-sub-tasks and so on.
- Group these as **plans** which specify how the tasks might be performed in practice

- Start with a user goal
  then identify main tasks for achieving it

Example Hierarchical Task Analysis

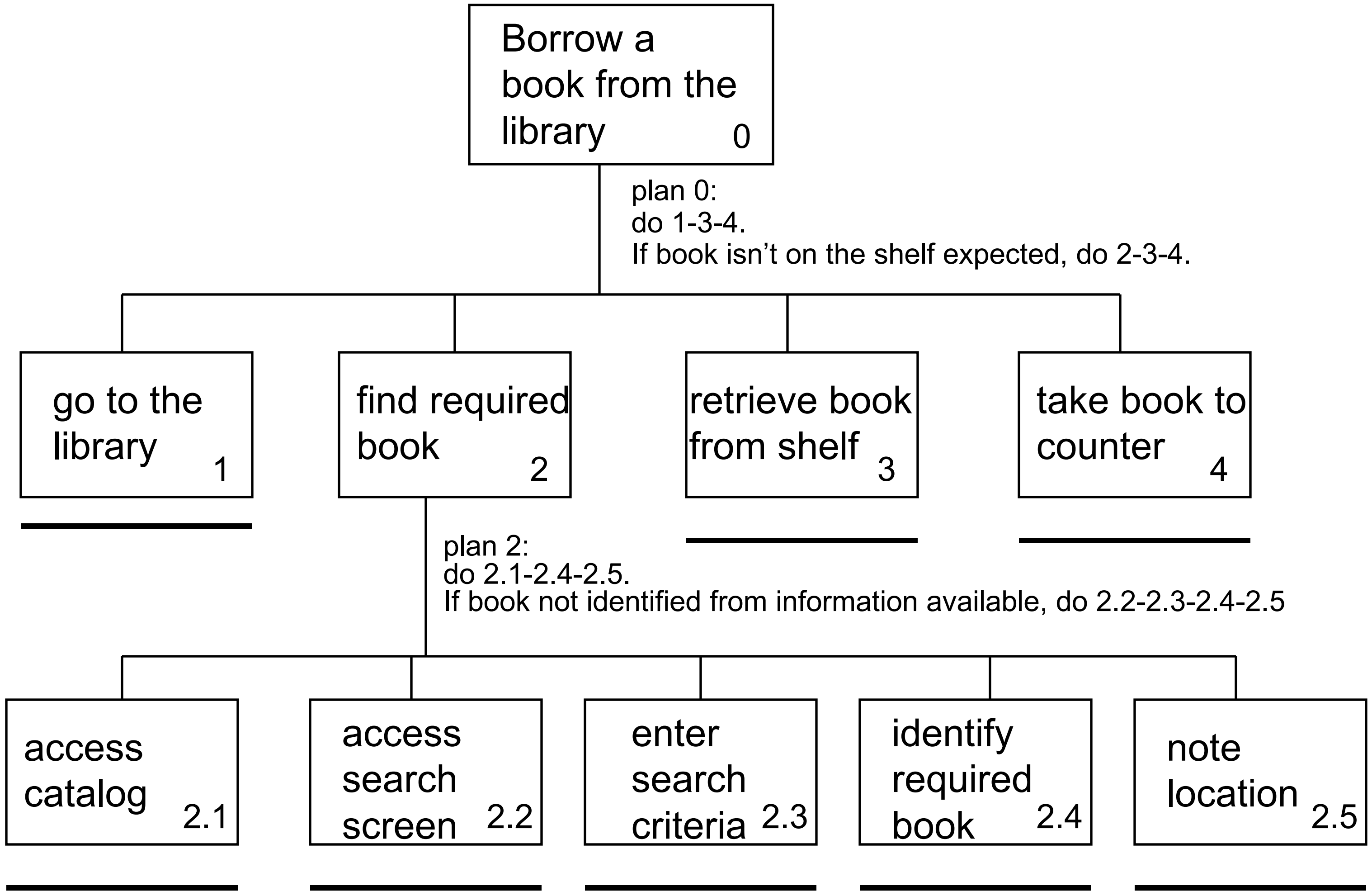
0. In order to borrow a book from the library

- **1. go to the library**
- **2. find the required book**
  - **2.1 access library catalog**
  - **2.2 access the search screen**
  - **2.3 enter search criteria**
  - **2.4 identify required book**
  - **2.5 note location**
- **3. go to correct shelf and retrieve book**
- **4. take book to checkout counter**

plan 0: do 1-3-4. If book isn’t on the shelf expected, do 2-3-4.

plan 2: do 2.1-2.4-2.5. If book not identified do 2.2-2.3-2.4.

Example Hierarchical Task Analysis (graphical)


Borrow a book from the library 0
plan 0:
do 1-3-4.
If book isn't on the shelf expected, do 2-3-4.
go to the library 1
find required book 2
retrieve book from shelf 3
take book to counter 4
plan 2:
do 2.1-2.4-2.5.
If book not identified from information available, do 2.2-2.3-2.4-2.5
access catalog 2.1
access search screen 2.2
enter search criteria 2.3
identify required book 2.4
note location 2.5

How Could This Be Useful?

- Manuals, training, help systems
- Requirements capture and system design
  - Models how the user would use the system
  - Based on existing system
    - What should be added? Where do new features fit?
    - What can be left out?
    - What’s most critical? What’s most frequently done?
  - May help you choose a high-level interaction style or think about a conceptual model
- Detailed interface design
  - Plans map to paths through dialogs
  - Menu design based on task decomposition
- Scenarios for user evaluation tests

Limitations of Task Analysis

- Not really a replacement for other methods that define <u>requirements for development</u>
- Is it ever complete?
- When to stop decomposing tasks?
- Task Analysis, HTA may not scale well
- Perhaps not best for
  - Overlapping, parallel tasks
  - Interruptions

Summary

- Getting requirements right is crucial
- There are different kinds of requirement, each is significant for interaction design
- The most commonly-used techniques for data gathering are: questionnaires, interviews, focus groups and workshops, naturalistic observation, studying documentation
- Scenarios, use cases and essential use cases can be used to articulate existing and envisioned work practices.
- Task analysis techniques such as HTA help to investigate existing systems and practices