Today’s Agenda

- Concept-level sequence diagrams (from last class)
- User interface requirements – two methods:
  - GUI-dependent specification
  - GUI-independent specification
- Validation: Usability tests

Example: Electronic Voting System (Sensus)

Readings: none

Note: see rearrangement of schedule on course web page.
Review

We discussed two methods for choosing a good OO decomposition for the system:

- Noun/verb analysis
- Using the system state diagram

State diagrams are needed for each non-trivial concept:

- All trigger events are operation calls (or “when” or “after”)
- No activities

The concept-level state diagrams show:

- When an operation on an object can be called.
- The effects of calling an operation on an object (change state, change attributes, return values).

Objects work together to accomplish the behaviour of the system state diagram (illustrated in the system sequence diagrams).
User Interfaces: a “How” Issue?

We used to think that specifying the user interface (UIs) for a system was a “how” issue rather than a “what” issue, meaning it should not be specified in the requirements specification and should be left to the implementers to decide.

However, we have had enough catastrophes in which the culprit was a poor UI interface on the controlling application that left the operator confused as to what was happening and he or she made a poor choice of what to do.

or

he or she made the wrong request.

So, now we specify the user interface as part of the requirements.
UI Requirements

We need to:

- specify the user interfaces, and

- often, validate proposed UIs with usability testing by real users (mock-up, prototype).

The UIs must be carefully designed along with the functional requirements to be consistent with the functional requirements.

Another good reason for specifying the UI is that it’s difficult to add them later to the code because the proper hooks may not have been left in the code, or we end up having to add our system’s function into a UI framework.
UI Requirements

In UI requirements, we want to describe:

- how input is provided to the system
- what the screens or windows will look like (or other kinds of outputs)
- when those screens or windows will appear (or other kinds of outputs)
- navigation among multiple screens

NOTE: This is NOT a lecture on what makes a good user interface!
Two Methods

Specifying the user interface can be done at two levels of detail:

1. **GUI-dependent**: integrate GUI events into the specification

2. **GUI-independent**: try to keep the aspects of the GUI independent from the rest of the specification as much as possible
GUI-Dependent

- Use the GUI events (i.e., GUI widgets – pressing a button on the screen including navigation buttons) as first-class events that are used in the use cases or system state diagrams (SSDs).

- Attach references to screen shots to use case steps or parts of the SSD.

- In the UI section, include the screen shots and describe the match between widgets and the GUI events.

- Navigation between screens is implicit in the use cases or SSDs.
GUI-Independent

- Don’t use GUI events or mention screen shots in use cases or SSDs.

- In the UI specification section:
  - Show the screen shots.
  - Associate the GUI events on the screen shots with inputs and outputs mentioned in the use cases and SSD.
  - Describe the behaviour of the navigation widgets (e.g., “ok” button moves the system to displaying screen shot #2).
  - Have a navigation diagram to show the order in which the system moves between screen shots. This must be consistent with the use cases and SSD!
Two Methods

Specifying the user interface can be done at two levels of detail:

1. **GUI-dependent**: integrate GUI events into the specification

2. **GUI-independent**: try to keep the aspects of the GUI independent from the rest of the specification as much as possible

Option #2 is recommended when possible because it keeps the UI independent (and therefore more easily changeable) from the rest of the specification.

Option #1 can sometimes be more clear (although more work) and avoids the issue of the navigation diagram being consistent with the SSD. The SSD will likely be more detailed in Option #1.
Example: Sensus

Here is a brief description of an electronic voting system called Sensus, due to Lorrie F. Cranor (Carnegie Mellon University).

Sensus has four main modules:

- **Registrar**: The Registrar registers voters prior to an election.

- **Validator**: The Validator ensures that only registered voters can vote, and that only one ballot is counted for each registered voter.

- **Tallier**: The Tallier tallies the results of the election or survey.
Example: Sensus

- **Pollster**: The Pollster acts as a voters’ [sic] agent, presenting human readable ballots to a voter, collecting the voter’s responses to ballot questions, performing cryptographic functions on the voter’s behalf, obtaining necessary validations and receipts, and delivering ballots to the ballot box.

The Pollster is the only component of the Sensus system that voters must trust completely; voters concerned about the privacy of their ballots may want to install personal copies of the pollster on trusted machines.

We will specify the behaviour of the pollster and consider the other components to be actors of the system.
Use Cases

The author of the *Sensus* description has provided us with two use cases:

- registering to vote
- marking a ballot (or voting)
GUI-Dependent UI Spec

- Describe the use cases (and SSD) in terms of GUI objects (buttons, menus, etc.).

- At each step in any use case that a particular screen is desired, mark the use case step with a numbered marker and then show pictures of these screens, each with its own unique numbered marker in the UI section. Several steps may share the same numbered marker, meaning that the same screen is used for each such step.

In addition to numbering, if in HTML or pdf, you can make each numbered marker a hot link.

- In the UI section, include the screen shots and the match between widgets and the GUI events.
# UC “Registering to Vote”

Precondition: The voter has obtained a voter [identification number, token, and registration address] from the election administrators.

<table>
<thead>
<tr>
<th>Voter «actor»</th>
<th>Pollster</th>
<th>Registrar «actor»</th>
</tr>
</thead>
</table>
| 1. Voter invokes the *sensus* command, to run the Pollster. | 2. Pollster displays a menu with options “register to vote” and “mark ballot”.

2. **1**

3. Voter selects the “register to vote” option. | 4. Pollster generates and shows a public/private key pair for Voter.

4. **2**

5. Voter hits “ok” button. | 6. Pollster prompts voter for his/her [identification number, token, and registration address].

6. **3**
<table>
<thead>
<tr>
<th>Voter «actor»</th>
<th>Pollster «actor»</th>
<th>Registrar «actor»</th>
</tr>
</thead>
<tbody>
<tr>
<td>7. Voter types his/her [identification number, token, and registration address].</td>
<td>9. Pollster prepares registration request on Voter’s behalf.</td>
<td>11. Registrar sends acknowledgement to Pollster within a few seconds.</td>
</tr>
<tr>
<td>8. Voter hits “ok” button.</td>
<td>10. Pollster submits registration request to registrar.</td>
<td></td>
</tr>
<tr>
<td>13. Voter sends a filename he/she will remember to Pollster; this name must be unique for the registration authority.</td>
<td>12. Pollster prompts Voter for a filename for saving his/her registration information.</td>
<td></td>
</tr>
<tr>
<td>Voter «actor»</td>
<td>Pollster «actor»</td>
<td>Registrar «actor»</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>14. Voter hits “ok” button.</td>
<td>15. If Voter has a .sensus directory, Pollster creates file with selected name in Voter’s .sensus directory and stores registration information in the file.</td>
<td>16. <em>sensus</em> program exits.</td>
</tr>
</tbody>
</table>
### EXCEPTION for step 11:

<table>
<thead>
<tr>
<th>Voter «actor»</th>
<th>Pollster «actor»</th>
<th>Registrar «actor»</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>11. Registrar does not send acknowledgement to Pollster, ...</td>
</tr>
</tbody>
</table>

### EXCEPTION for step 15:

<table>
<thead>
<tr>
<th>Voter «actor»</th>
<th>Pollster «actor»</th>
<th>Registrar «actor»</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>15. If Voter does not have a .sensus directory, Pollster creates one.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16. Pollster creates file with selected name in Voter’s .sensus directory and stores registration information in the file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17. sensus program exits.</td>
</tr>
</tbody>
</table>
## UC “Marking your Ballot”

<table>
<thead>
<tr>
<th>Voter «actor»</th>
<th>Pollster</th>
<th>Election «actor»</th>
<th>Validator «actor»</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1.</strong> Voter invokes the <em>sensus</em> command, to run the Pollster.</td>
<td>2. Pollster displays a menu with options “register to vote” and “mark ballot”. ¹</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.</strong> Voter selects the “marking ballot” option.</td>
<td>4. Pollster generates and shows a public/private key pair for Voter. ²</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>5.</strong> Voter hits “ok” button.</td>
<td>6. Pollster prompts voter for his/her [identification number, token, and registration address]. ³</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Voter «actor»</td>
<td>Pollster</td>
<td>Election «actor»</td>
<td>Validator «actor»</td>
</tr>
<tr>
<td>----------------</td>
<td>----------</td>
<td>------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>7. Voter types his/her [identification number, token, and registration address].</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. Voter hits “ok” button.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10. Validator sends Pollster acknowledgement that Voter is registered.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11. Pollster requests an unvoted ballot from Election authority to send it to Voter.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12. Election authority sends an unvoted ballot to Pollster</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Voter «actor»</td>
<td>Pollster</td>
<td>Election «actor»</td>
<td>Validator «actor»</td>
</tr>
<tr>
<td>--------------</td>
<td>----------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>14. Voter hits “ok” button.</td>
<td>13. Pollster tells Voter unvoted ballot is available.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16. Voter selects “view ballot questions and instructions” from the pollster menu.</td>
<td>15. Pollster displays pollster menu.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18. Voter chooses to return to pollster menu.</td>
<td>17. Pollster displays ballot questions and instructions.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20. Voter selects “mark ballot” from the pollster menu.</td>
<td>19. Pollster displays pollster menu.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Voter «actor»</td>
<td>Pollster «actor»</td>
<td>Election «actor»</td>
<td>Validator «actor»</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------</td>
<td>------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>22. Voter types the name of the ballot and Voter's registration file name.</td>
<td>21. Pollster prompts for the name of the ballot and Voter's registration file name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23. Voter hits “ok” button.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>24. Pollster displays ballot question 1 and its voting instructions.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26. Voter hits “ok” button.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31. Voter votes on ballot question m.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32. Voter hits “ok” button.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Voter «actor»</td>
<td>Pollster «actor»</td>
<td>Election «actor»</td>
<td>Validator «actor»</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------</td>
<td>------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>33. Pollster prompts Voter with vote-ending prompt 1.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34. Voter answers “yes” to prompt 1.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35. Voter hits “ok” button.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36. Pollster prompts Voter with vote-ending prompt $m$ and informs Voter.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37. Voter answers “yes” to prompt $m$.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38. Voter hits “ok” button.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Voter «actor»</td>
<td>Pollster «actor»</td>
<td>Election «actor»</td>
<td>Validator «actor»</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td></td>
<td>39. Pollster sends completely voted ballot to Election Authority.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40. Voter hits “ok” button.</td>
<td>41. <em>sensus</em> program exits.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ALTERNATIVE for steps 34–38 for Voter that does not want to review the ballot before marking it:

<table>
<thead>
<tr>
<th>Voter</th>
<th>Pollster</th>
<th>Election</th>
<th>Validator</th>
</tr>
</thead>
<tbody>
<tr>
<td>«actor»</td>
<td>«actor»</td>
<td>«actor»</td>
<td>«actor»</td>
</tr>
</tbody>
</table>

After step 15 comes step 20.

ALTERNATIVE for steps 25–32 for Voter that quits after some number of ballot questions and goes back to remark the entire ballot, because he or she has changed his or her mind or has made a mistake:

<table>
<thead>
<tr>
<th>Voter</th>
<th>Pollster</th>
<th>Election</th>
<th>Validator</th>
</tr>
</thead>
<tbody>
<tr>
<td>«actor»</td>
<td>«actor»</td>
<td>«actor»</td>
<td>«actor»</td>
</tr>
</tbody>
</table>

25. Voter hits “cancel” button to cancel voting.

Go to step 15.
### ALTERNATIVE for steps 34–38:

<table>
<thead>
<tr>
<th>Voter «actor»</th>
<th>Pollster «actor»</th>
<th>Election «actor»</th>
<th>Validator «actor»</th>
</tr>
</thead>
</table>

34. Voter answers “no” to a vote-ending prompt $i$, where $i < m$
   Go to step 15

### EXCEPTION for step 10:

<table>
<thead>
<tr>
<th>Voter «actor»</th>
<th>Pollster «actor»</th>
<th>Election «actor»</th>
<th>Validator «actor»</th>
</tr>
</thead>
</table>

10. Validator sends voterId is not registered ...
Annotating SSDs

Use cases are not specifications!

They only illustrate functionality from the user’s point of view.

Thus a UI specification tied to use cases is not really a specification, and may not show some options.

Therefore, it might be better to annotate the SSD, than the use cases!

Associate screen shot numbers with states or transitions in your SSD. (These should correspond to the use case step that had that annotation.)

You may need to indicate which transition is taken in response to any textual input or to any widget selection or clicking.
Example
Screen Pictures

The screen pictures follow this slide.
Menu of Options

Select one option:

☐ Register to vote
☐ Mark ballot
Your public/private key pair is:

555555555555555555555555555555/666666666666666666666666666666

Don’t forget it!
Registration Prompt 3

Please enter your

Voter identification no.  

Voter token  

Registration address  

OK  CANCEL
Please enter a file name for saving your registration information.
Your pollster has just been sent an unvoted ballot.

You may now choose to mark your ballot.
Pollster Menu 6

Select one command:

☐ View ballot questions and instructions
☐ Mark ballot
View Ballot 7

Issue/Question/Office 1

☐ Choice 1
☐ Choice 2 Vote for only 1 choice

☐ Choice j

Issue/Question/Office 2

☐ Choice 1

Continued next page
View Ballot, Scrolled (7 con’d)

Issue/Question/Office 4

- Choice 1
- Choice 2

Vote for at most 3 choices

- Choice h

return to pollster menu
Please enter the name of the ballot for which you wish to vote.

Please enter the name of your registration file, found in your .sensus directory.

OK

CANCEL
Display Ballot Question 9

Issue/Question/Office x

- Choice 1
- Choice 2
- Choice j

Vote for only 1 choice

Return to pollster menu

OK  CANCEL
Vote Ending Question 10

Vote ending question i

Choose exactly 1 choice

Yes
No

OK  CANCEL
Voted Ballot Sent 11

Your voted ballot has been sent to the election authority.

Answering OK will exit sensus.
Section 4: Interface Requirements

4.1 User Interfaces

- drawings of windows
- purpose of each UI widget (button, menu option, etc.) in window:
  - link between user interface widgets and input events (e.g., we need to be able to tell which widget on which dialog can cause the “createCustomer” input event)
  - effect of output events on window display (1-2 line description)
- how to navigate among windows

A 1–2 line description suffices to describe a purpose or an effect. Navigation information may already be covered under descriptions of purposes (e.g., of a menu option).

You may choose your own format for the above parts.
Example: Screen Shot 2

Your public/private key pair is:
555555555555555555555555555555/
666666666666666666666666666666
Don’t forget it!

Public key (output text): Voter’s public key

Private key (output text): Voter’s private key

Ok button (input button): moves the system to screen shot 3

Cancel button (input button): move the system to screen shot 1
Example: Screen Shot 3

Please enter your
Voter identification no.  
Voter token  
Registration address  

OK  CANCEL

Voter Id no. text box (*input text box*): inputs voter id no.
Voter Token text box (*input text box*): inputs voter token
Registration address text box (*input menu*): inputs voter registration address from a list of possible addresses
Ok button (*input button*): moves the system to screen shot 4 (if registering to vote) or screen shot 5 (if marking ballot).
Cancel button (*input button*): move the system to screen shot 1
GUI-Independent

- Don’t use GUI events or mention screen shots in use cases or SSDs.

- In the UI specification section:
  - Show the screen shots.
  - Associate the GUI events on the screen shots with inputs and outputs mentioned in the use cases and SSD.
  - Have a navigation diagram to show the order in which the system moves between screen shots. This must be consistent with the use cases and SSD!
  - Describe the behaviour of inputs that are for navigation only (e.g., “ok” button moves the system to displaying screen shot #2)
Section 4: Interface Requirements

For the GUI-independent approach, Section 4 is very similar to what it is for the GUI-dependent approach.

The difference is that there will be a less obvious relationship between the widgets and the input/outputs of the SSD because the SSD does not refer directly to the GUI events.
Navigation Diagram

Purpose: show the order of presentation of screen shots. (In the GUI-dependent approach, this is given in the SSD annotated with screen shots.)

This information must be consistent with the descriptions provided for the navigation buttons.
User Interface Specs

- Screen shots may be created using any method you like.
- Don’t get caught up in fancy features of a UI (e.g., colours, position on screen, type of button).

The screen shots in a UI specification are intended to show when different inputs are required and the form of the input (e.g., textbox, button, etc.)

Style will be added later.
Validation: Usability Tests

- Prototype
- Mockups (may be entirely on paper – use hands to point at buttons)
  - Log keeper notes which problems user encounters
  - Have user think aloud (not for time keeping)
  - Can do early during requirements engineering
- Heuristic evaluation (HCI expert)
Summary

- Concept-level sequence diagrams (from last class)
- User interface requirements – two methods:
  - GUI-dependent specification
  - GUI-independent specification
- Validation: Usability tests

Example: Electronic Voting System (Sensus)

Next Lecture: Review for midterm exam

Reading: none