

Lecture 11
Project stuff
User interface design

Weekly TA meetings start March 25

- **one grad TA will be your advisor / mentor**
 - advise and monitor so you don't get stuck
 - help find other groups that have already overcome your current hurdles if possible
- **everyone comes to each meeting prepared**
 - what progress have you made, what next steps are planned, what problems do you need help with
- **it's your project**
 - It's your project to design and scope
 - It's your code to write and debug
 - It's your application to put into the hands of users to test

CAS: Central Authentication Service

- if your project requires users to log in with a Princeton netid, don't ask them to send passwords directly



Help

Central Authentication Service

A screenshot of the Central Authentication Service login page. It features two input fields: "NetID" and "Password". To the right of the "Password" field is a padlock icon. Below the input fields is an orange "LOGIN" button.

NetID

Password

LOGIN

- OIT provides a central authentication service

- the user visits your startup page
- the user is asked to authenticate via OIT's service
- the name and password are sent to an OIT site for validation without passing through your code at all
- if OIT authenticates the user, your code is called

- OIT web page about CAS:

`https://sp.princeton.edu/oit/eis/iam/authentication/CAS/CAS%20Developer%20KB.aspx`

- sample code is on course web page

Authentication for projects (etc.)

- PHP version

```
<?php
require 'CASClient.php';
$C = new CASClient();
$netid = $C->Authenticate();
echo "Hello $netid"; // or other code
?>
```

- Python version

```
import CASClient, os
C = CASClient.CASClient()
netid = C.Authenticate()
print "Content-Type: text/html\n"
print "Hello %s" % netid # or other code
```

- Java version

```
CASClient casClient = new CASClient();
String netid = casClient.authenticate();
System.out.println("Content-type: Text/html\n");
System.out.println("Hello " + netid);
```

Behind the scenes in the client libraries

- your web page sends user to

```
https://fed.princeton.edu/cas/login?  
service=url-where-user-will-log-in
```

- CAS sends user back to the service url to log in
with a parameter `ticket=hash-of-something`

- your login code sends this back to

```
https://fed.princeton.edu/cas/validate?  
ticket=hash&service=url...log-in
```

- result from this is either 1 line with "no"
or two lines with "yes" and *netid*

Source code management systems

- Git, SVN, Mercurial, Bazaar, Perforce, ...
- for managing large projects ~~with multiple people~~
 - works locally or across a network
- store and retrieve all versions of all directories and files in a project
 - source code, documentation, tests, binaries, ...
- support multiple concurrent users
 - independent editing of files
 - merged into single version
- **a Github repository is required for COS 333 projects!**
 - saves all previous versions of all files so you can back out of a bad change
 - logs changes to files so you can see who changed what and why
 - maintains consistency by resolving conflicting changes made by different users
 - protects against disasters

Basic sequence for all systems

- **create a repository that holds copies of your files**
 - including all changes and bookkeeping info
- **each person checks out a copy of the files**
 - "copy - modify - merge" model
 - get files from repository to work on
 - does not lock the repository
 - make changes in a local copy
 - when satisfied, check in (== commit) changes
- **if my changes don't conflict with your changes**
 - system updates its copies with the revised versions
 - automatically merges edits on different lines
 - keeps previous copies
- **if my changes conflict with your changes**
 - e.g., we both changed lines in the same part of a file,
 - checkin is not permitted
 - we have to resolve the conflict manually

Basic Git sequences (git-scm.com/documentation, gitref.org)

```
cd project
```

```
git init
```

makes .git repository

```
git add .
```

```
git commit
```

makes a snapshot of current state

[modify files]

```
git add ... [for new ones]
```

```
git rm ... [for dead ones]
```

```
git commit
```

```
git log --stat
```

```
git clone [url]
```

makes a copy of a repository

xkcd.com/1597

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.



xkcd.com/1296





	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Private Github repositories (free)

- <https://researchcomputing.princeton.edu/faq/how-do-github-permissions/>

Yes! Recently GitHub.com changed the way organizations grant permissions to repositories. Existing repositories, and access to them are unaffected. However, you will now be able to control repository access yourself on a per-repo basis, create teams, and add/remove members from a team. You will now need to request membership in the organization, and request the creation of private repositories using the [GitHub Service Request form](#) .

Fill out the [GitHub Service Request form](#)  with your GitHub username, and a unique name for a repo. We will create a private repository and transfer administrative control to you.

User interface design

"The interface is the product."

Jef Raskin

originator of the Apple Macintosh

"A user interface is well designed when the program behaves exactly how the user thought it would."

Joel Spolsky

User Interface Design for Programmers

Some of this material is ~~stolen~~ borrowed

Marshini Chetty

Princeton University

Research Field:

Human Computer Interaction (HCI)

Areas of Interest:

Ubiquitous Computing

Usable Privacy and Security

Information and Communication

Technology for Development (ICTD)

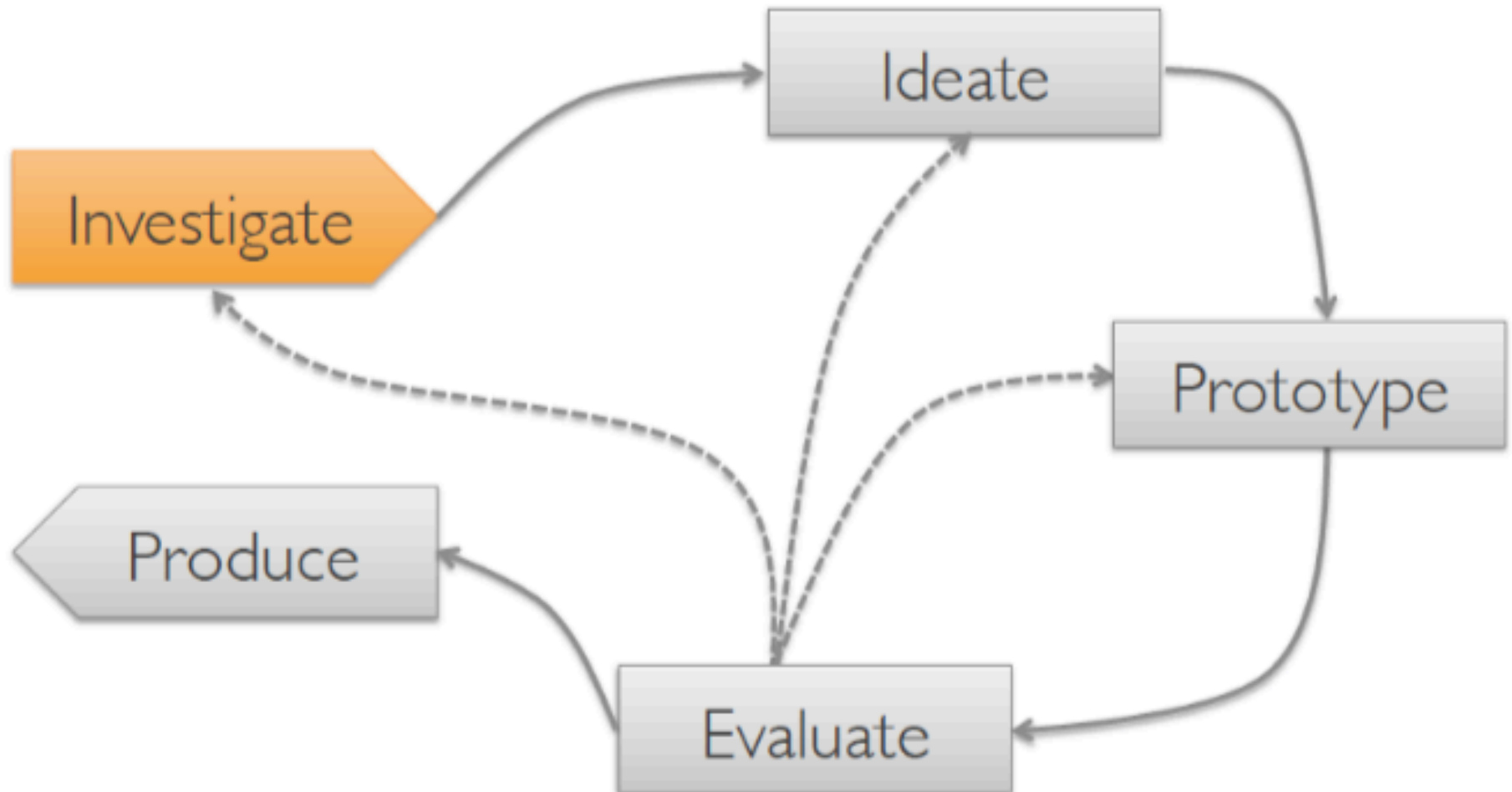
Contact:

marshini@princeton.edu

CS 412



User-centered design process



Investigate

- who are the users and other interested parties?
- what do they do now?
- what do they want to do?
- what has been tried already?

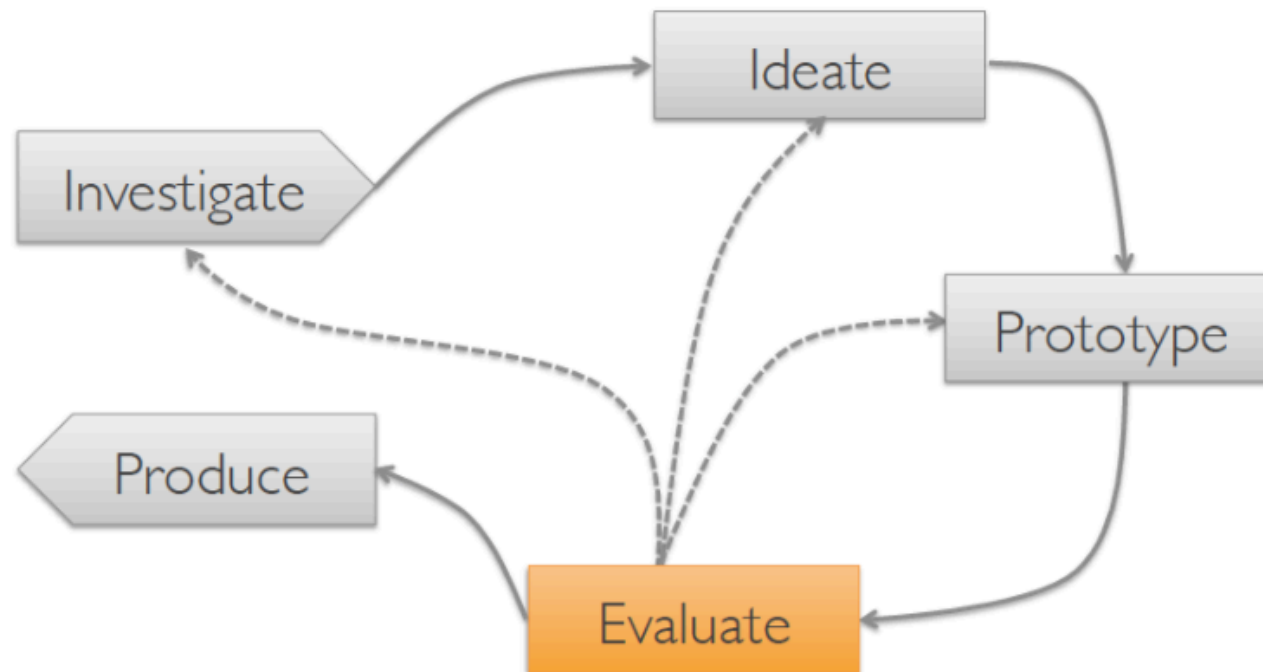
- user surveys
- interviews
- focus groups
- competitive analysis
- ...

Prototype

- hard to evaluate something if it doesn't exist
 - hard for users to react to abstract concepts
 - prototype reveals things that were not thought of originally
 - start to see technical issues arising
-
- low and medium fidelity:
 - build fast
 - don't over-engineer
 - don't become attached to it
 - build more than one at the same time

Evaluate

- walk through prototypes
 - think-aloud
 - usability tests and experiments
 - A/B testing
-
- feed this back into the cycle at various places

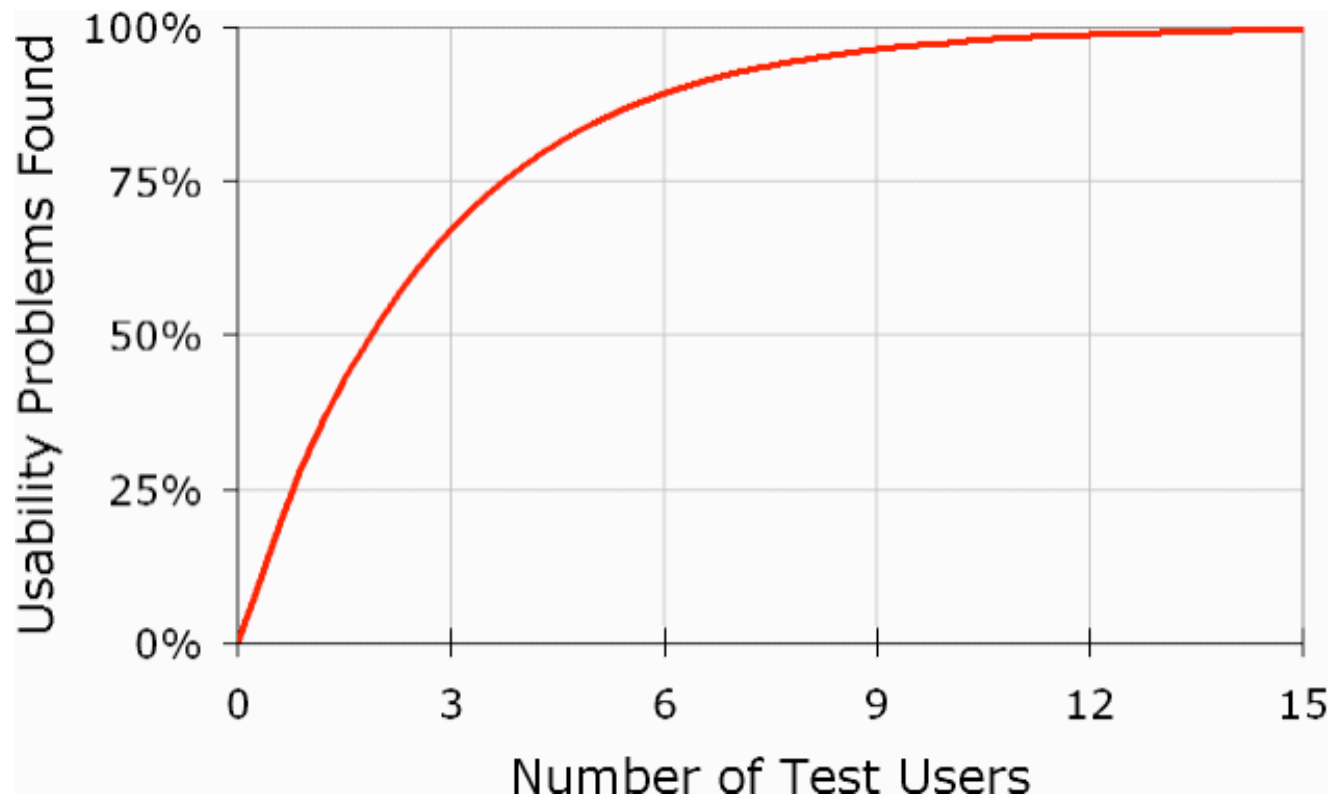


Usability testing (Krug, *Rocket Surgery Made Easy*)

- figure out a small set of important tasks
- make scenarios that tell the user what to perform
 - but not how to do it
- try it yourself
- write it down so the user can refer to it
- get the user to talk out loud about his/her thought processes while performing it

Get users early

- "Elaborate usability tests are a waste of resources. The best results come from testing no more than 5 users and running as many small tests as you can afford."



<https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/>

Uniformity, consistency

- **uniform visual appearance throughout: sizes, colors, fonts, ...**
 - CSS is your friend for web pages
- **always put similar items in standard places**
 - File / Edit / View ... / Help
 - OK / Cancel at the bottom right; Do Nothing at bottom left, separated
 - consistency with other systems if it makes sense
 - use the local look and feel
- **follow conventions**
 - ... means "more to come"
 - grayed out means inactive
 - checkmark means currently selected
 - ...

Legibility

- not everyone is 18 years old!
- use large enough text sizes
- use legible fonts
- use contrasting colors
 - don't use dark blue letters on black background!
- don't assume a screen size
 - text should adapt to screen size, not require scrolling
- mobile is different but the same principles apply

Ease of use

- pick good defaults
- don't get too deep
 - multi-level menus
 - drop-downs that don't fit on the screen
- remember useful information
 - previous text entries
 - position in file system
 - settings
- but do it consistently
- provide text-based alternatives
 - let me type filenames instead of forcing a dialog
- think about accessibility
- think about other languages and cultures

Safety first

- **don't do irrevocable actions without confirmation**
 - don't quit without warning if changes are not saved
- **but do it right**
 - don't ask about saving if there were no changes
 - Excel and Word both do this wrong
 - show an indicator of whether changes have occurred
 - don't make changes if there were none!
- **provide a way to interrupt long-running computations safely**
- **watch out for security warnings that are clicked automatically**

Useful reading

- **Windows user experience guidelines:**
 - <https://docs.microsoft.com/en-us/windows/desktop/uxguide/guidelines>
 - <https://docs.microsoft.com/en-us/windows/desktop/appuistart/-user-interface-principles>
- <http://www.nngroup.com/articles/>
 - (Don Norman, Jakob Nielsen, Bruce Tognazzini, ...)
- goodui.org
- Steve Krug, *Rocket Surgery Made Easy*
- Joel Spolsky, *User Interface Design for Programmers*
- World's worst web site (two of a million similar sites)
 - <http://www.angelfire.com/super/badwebs/>
 - <http://www.webpagesthatsuck.com/> (not updated)