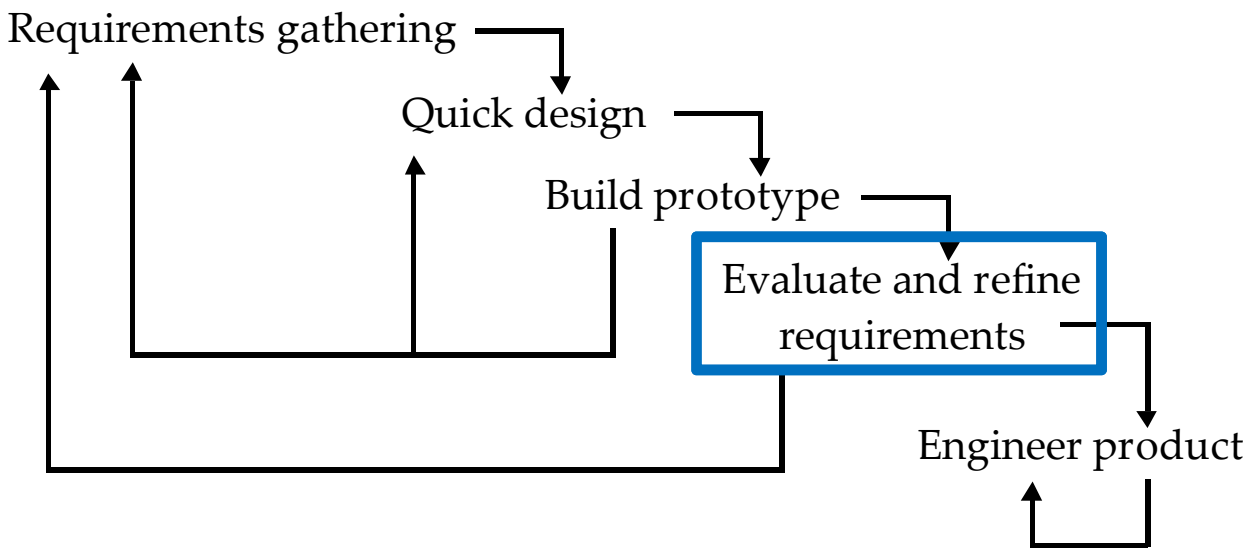
Interaction Design

Heuristic Evaluation & Cognitive Walkthrough

Interaction Design

- **Iterative** user centered design and development



Recap: Usability

- Ease of learning
  - Faster the second time
- Recall
  - Remember how to use it next time
- Productivity
  - Perform tasks quickly and efficiently
- Minimal error rates
  - If they do occur, good feedback is given so that the user can recover
- High user satisfaction
  - Confident of success

Discount Usability Engineering

- A cost-effective method of **usability evaluation** that requires **fewer resources** and **time** than formal usability testing
  - Cheap
    - No special labs or equipment needed
    - The more careful you are, the cheaper it is
  - Easy to learn
    - Can be taught in 2-4 hours
  - Fast
    - Around a day

Discount Usability Engineering

- Lo-fi prototyping
  (looked at previously)
- Heuristic Evaluation
  - Ten usability heuristics
  - Severity ratings
  - Performing HE
  - Examples
- Cognitive Walkthrough

  Based on *The Cognitive Walkthrough: A Practitioner's Guide*" by C. Wharton, J. Rieman, C. Lewis, and P. Polson, U. of Colorado, Boulder

Heuristic Evaluation

Heuristic Evaluation

- Developed by Jakob Nielsen (from NeilsenNorman group)
- Helps **find usability problems** in a user interface (UI) design
  - In already built UI, or sketches
- Small set of evaluators used to examine UI (3-5)
  - Each checks for compliance with usability principles - *heuristics*
  - Use multiple evaluators as each will identify different problems
  - At the end of the session problems are compiled and used to inform re-design

Ten Usability Heuristics

H.1 visibility of system status
H.2 match between system and real world
H.3 user control and freedom
H.4 consistency and standards
H.5 error prevention
H.6 recognition rather than recall
H.7 flexibility and efficiency of use
H.8 aesthetic and minimalist design
H.9 help users recognize and recover from errors
H.10 help and documentation

Performing Heuristic Evaluation

1. Pre-evaluation training
   - Get evaluators up to speed on **domain** and **scenarios** to be used
2. Evaluate
   - Evaluators individually use UI according to scenarios
   - Go through twice
     - Once for overview, second time for detail
3. Collate results
4. Rate severity
5. Feedback into design

Severity Ratings (1)

- Combination of
  - Frequency of problem
  - Persistence of problem
  - Impact of problem
- Calculate after evaluations are complete
  - Each evaluator rates each problem
- Provides an indication of **the need** for more assessment and/ or **redesign**

Severity Ratings (2)

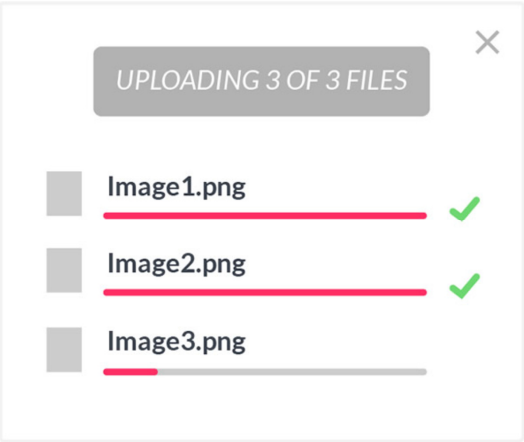
- 0 – do not agree that it is a usability problem
- 1 – it is a cosmetic problem
- 2 – minor usability problem
- 3 – major usability problem
  - important to fix
- 4 – usability catastrophe
  - imperative to fix

Ten Usability Heuristics

H.1 visibility of system status
H.2 match between system and real world
H.3 user control and freedom
H.4 consistency & standards
H.5 error prevention
H.6 recognition rather than recall
H.7 flexibility and efficiency of use
H.8 aesthetic and minimalist design
H.9 help users recognize and recover from errors
H.10 help and documentation

H.1 Visibility of System Status

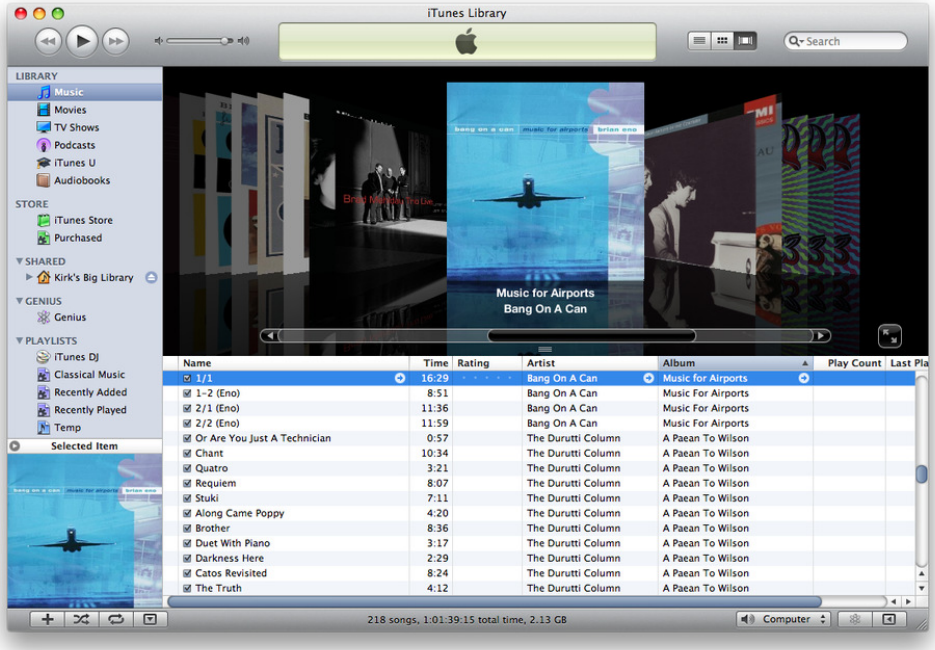
- Keep users informed about what is going on through appropriate feedback within reasonable time
  - Pay attention to response time
    - 0.1 sec: no special indicators needed
    - 1.0 sec: user tends to lose track of data
    - 10 sec: max. duration for the user to stay focused on 1 action
    - for longer delays use progress bars / indicators



H.2 Match to Real World (1)

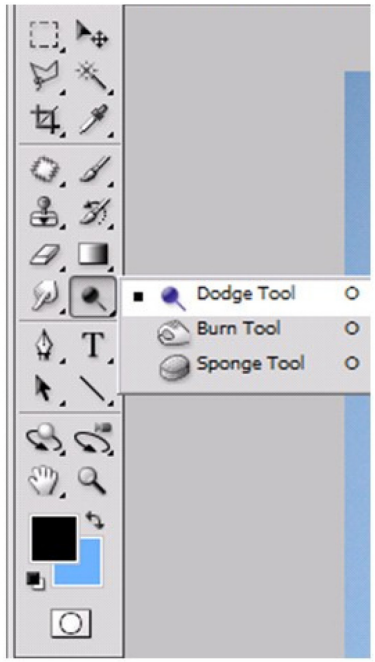
- Speak the users' language
- Follow real world conventions

  - **Example**: iTunes



H.2 Match to Real World (2)

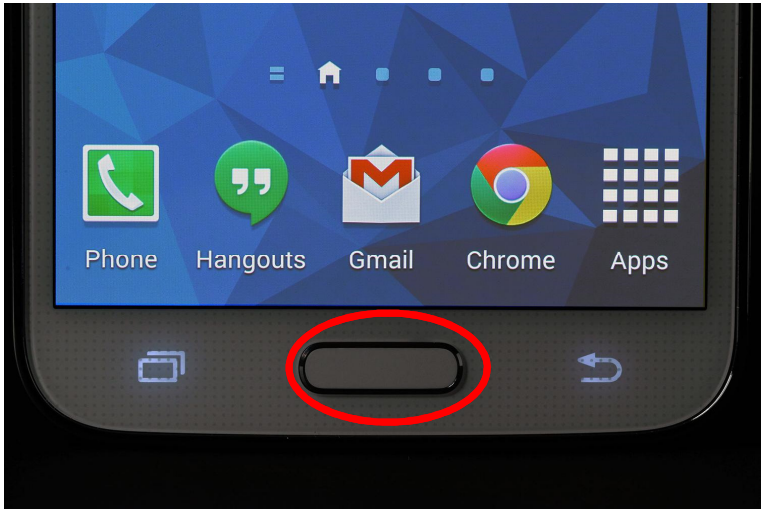
- Speak the users' language
- Follow real world conventions
  - **Example**: Photoshop



H.3 User Control and Freedom

- Clearly marked “exits” for mistaken choices
  - undo/ redo
- Do not force down fixed paths

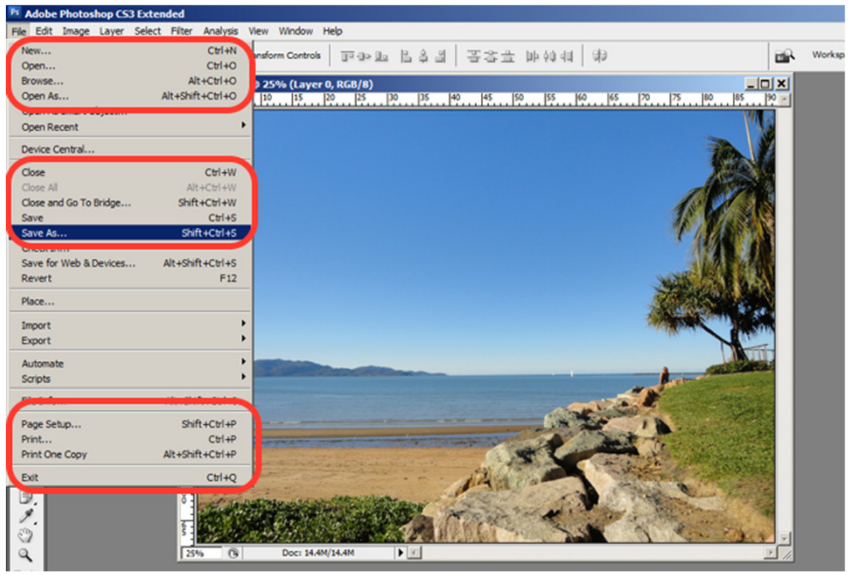
Example: Home button



H.4 Consistency & Standards

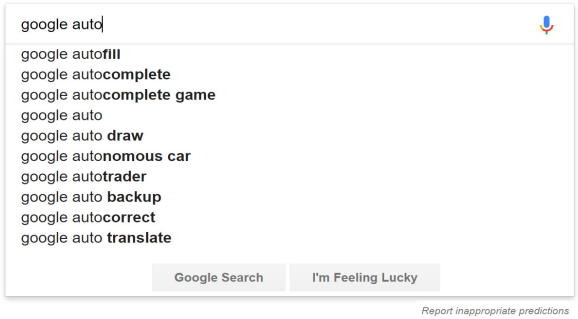
- Consistency within **and** between applications
  - e.g. Word, Excel, and PowerPoint all use the same style toolbar with the same primary menu options

Example: Adobe Photoshop



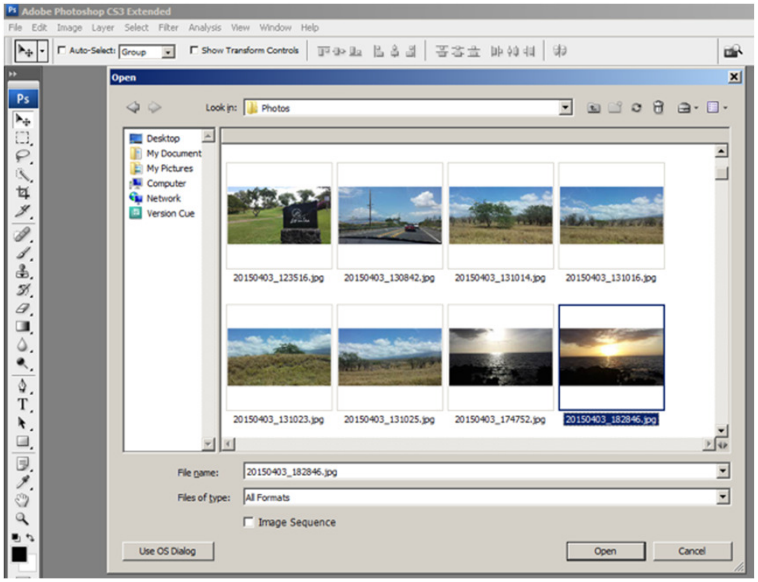
H.5 Error Prevention

- What is better than good error messages is a careful design, which prevents a problem from occurring in the first place
  - Example: if PIN is 4 digits, only allow 4 numeric characters
  - Example: **Google Autocomplete**



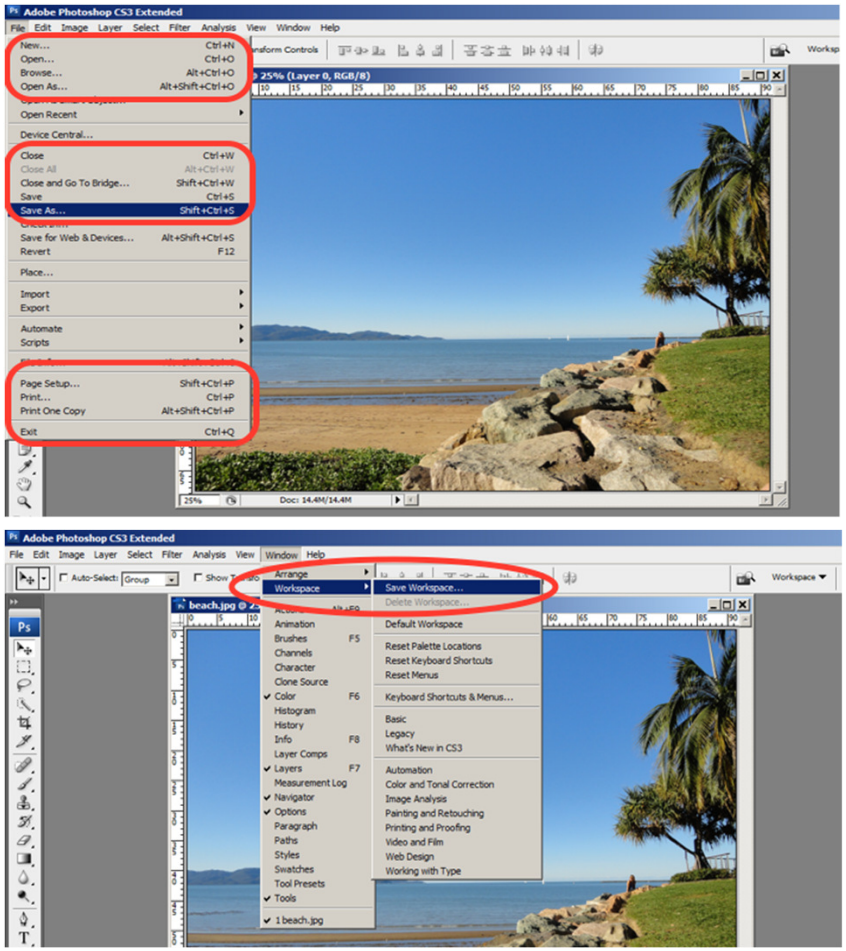
H.6 Recognition Rather than Recall

- Minimize user's memory load
- Make objects, actions, options, and directions visible or easily retrievable
  - Example: **Adobe Photoshop**



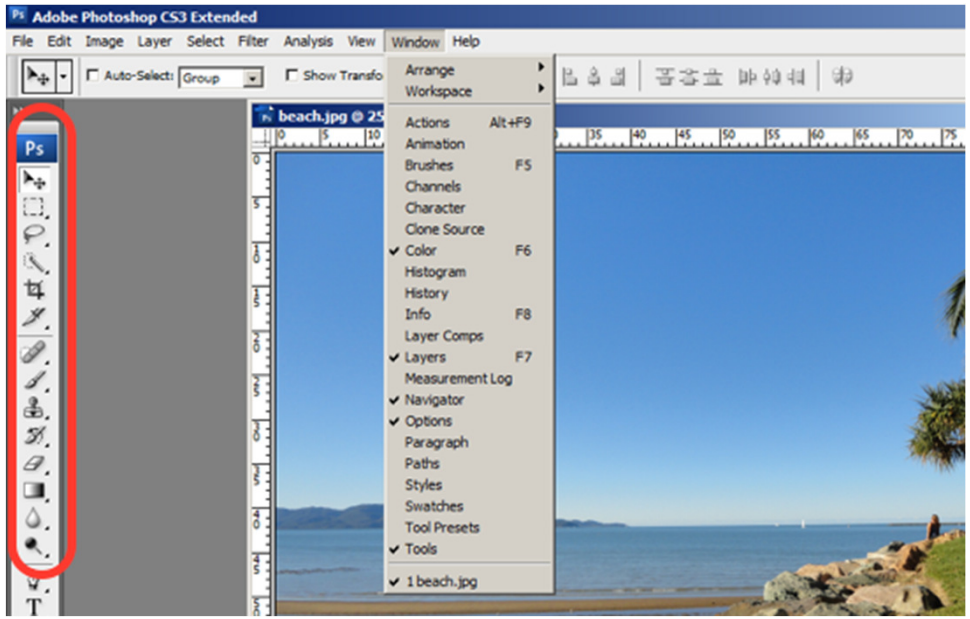
H.7 Flexibility for Efficient Use

- Accelerators for experts
  - e.g., gestures, keyboard shortcuts
- Allow users to tailor frequent actions
  - e.g., macros
- Support frequent tasks and tasks with high cognitive load
  - e.g., copy & paste



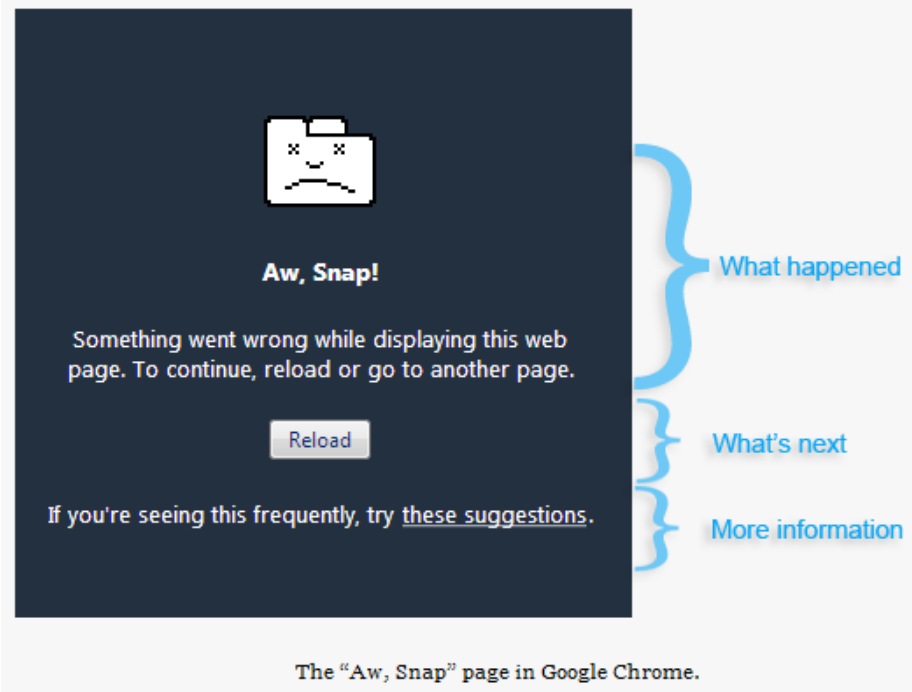
H.8 Aesthetic and Minimalist Design

- Draw the user to **focus on** the **main subject** at hand
- Keep the information displayed on the application **simple**
- **Categorize** repetitive information into relevant sections



H.9 Help Users Recover from Errors

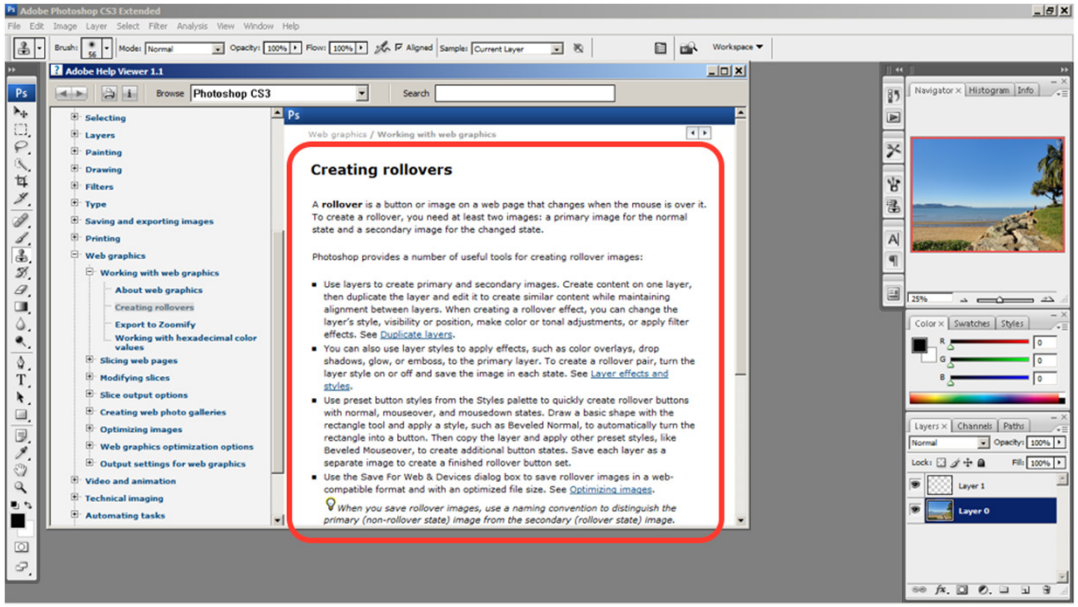
- Help users recognize, diagnose, and recover from errors
  - error messages in plain language
  - precisely indicate the problem
  - constructively suggest a solution



The "Aw, Snap" page in Google Chrome.

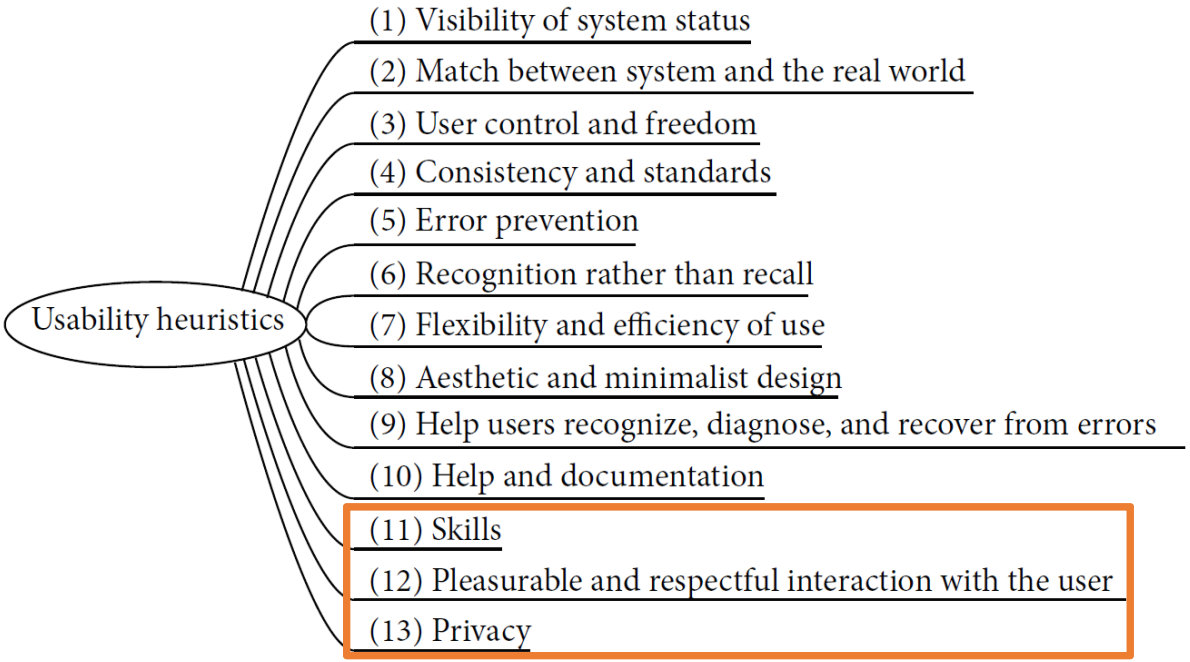
H.10 Help and Documentation

- Easy to search
- Focused on the user's task (contextual help)
- List concrete steps to carry out
- Not too long



Heuristic Evaluation: Current Trend

- Heuristic evaluation on **mobile interfaces**



Further Reading:
Gómez, Caballero, Sevillano, 'Heuristic Evaluation on Mobile Interfaces: A New Checklist', 2014.

Heuristic Evaluation : Example (1)

- **Cannot copy info from one window to another**
  - Which Heuristics are violated?
  - How can the problem be fixed?

Heuristic Evaluation : Example (2)

- **Typography uses mix of upper/lower case formats and fonts**
  - Which Heuristics are violated?
  - How can the problem be fixed?

Heuristic Evaluation : Example (3)

The interface used the string "Save" on the first screen for saving the user's file but used the string "Write file" on the second screen.

- Which Heuristics are violated?
- What is the severity rating?
  - Why?
    - What is the effect on the user?

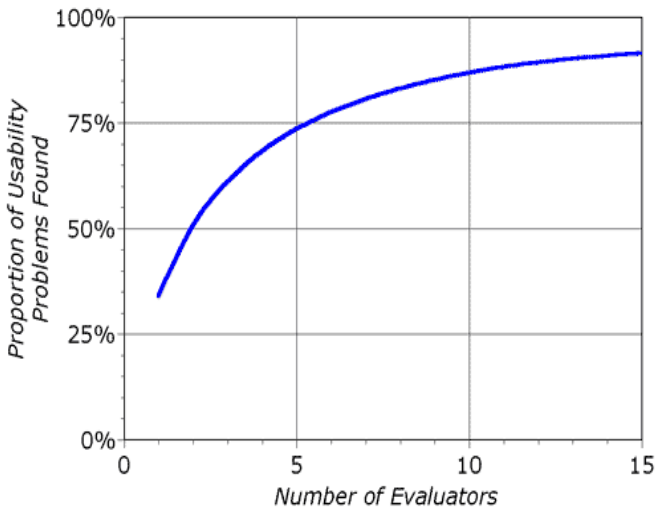
Results of Heuristic Evaluation

- Discount: benefit-cost ratio of 48 [Nielsen94]
  - cost was $10,500 for benefit of $500,000
  - how might we calculate this value?
    - in-house –> productivity; open market –> sales
- Single evaluator achieves poor results
  - only finds 35% of usability problems
  - 5 evaluators find ~ 75% of usability problems
  - why not more evaluators?

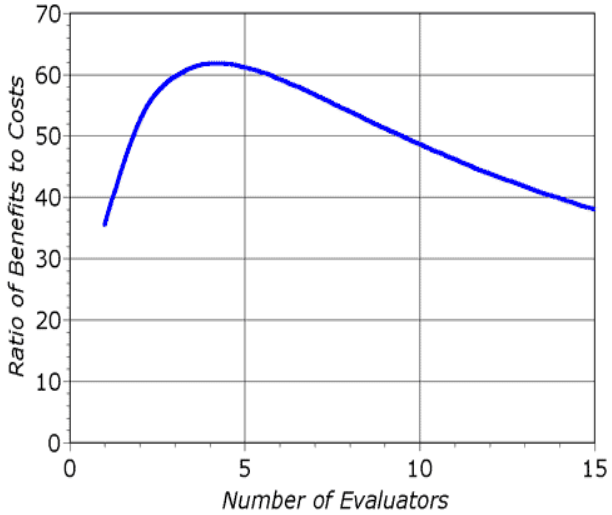
Diminishing Returns

- Adding evaluators costs more
- Many evaluators will not find many more problems

problems found



benefits / cost



HE Summary

- Cheap & quick
- Easy to learn
- Finds a lot of problems

Drawbacks

- Not task focused
- Not using actual people
- Not rigorous

Cognitive Walkthough

Cognitive Walkthough (CW)

- Cognitive walkthrough is a **task-centered evaluation**
- Focuses on real, complete, and representative tasks

Cognitive Walkthough: Why?

- Questioning **assumptions** about what users will be **thinking**
- Identifying **controls** that may be **missing** or hard to find
- Finding places that have **inadequate feedback**
- Suggesting **difficulties** users may have with **labels** or **prompts**

Cognitive Walkthough: Purpose

- Focuses on problems that users will have when they **first use** an interface, without training

- Most effective if designers can really create a **mental picture** of the actual environment **of use**

Cognitive Walkthough: Caveat

- Not a technique for evaluating the system over time
  - e.g., how quickly a user moves from beginner to intermediate

Preparation

Prior to doing a walkthrough, you need four things:

1. A description or a **prototype** of the interface
2. A **task description** for a representative task
3. A complete **list of the actions** needed to complete the task
4. An idea of who the **users** will be and what kind of **experience** they will bring to the job

CW: Step-by-step Guide

1. Define inputs
2. Get analysts
3. Step through action sequences for each task
4. Record important information
5. Revise user interface (UI)

Step 1. Define Inputs

- Define inputs by answering the following:
  - Who are the users?
  - What are the tasks?
  - What are the action sequences for the tasks?
- Have the prototype/ implementation / description of the interface

Step 2. Get Analysts

- You do not need actual users
- You can evaluate the interface by imagining the behaviour of entire classes of users, not one unique user
- A typical developer can perform cognitive walkthrough
  - But they should have a knowledge of cognitive science to understand people's limitations

Step 3. Step Through Actions

- Will users know what to do?
- Will users see how to do it?
- Will users understand from the feedback whether their actions are correct or not?

Step 4. Record Important Information

- User knowledge (just before and just after the action)
- Assumptions about users
- Side issues and design changes
- Credible success or failure story
  - i.e., why would a user select or not select the correct action?

Step 5. Revise the UI (1)

- If the user fails to select the right action
  - eliminate that action
  - prompt user for action
  - change other part so user knows that s(he) can try the action
- If the user does not know that the correct action is available
  - make action more obvious

Step 5. Revise the UI (2)

- If the user does not know which action is correct
  - label controls based on knowledge of users
  - check the sequence of actions - does it make sense?

- If the user can't tell things are going ok
  - give the user feedback, say what happened

Problems CW Finds

- **Severe problems** - Fairly good, comparable to other techniques
- **Content-related problems** - Comparable for consistency, worse for recurrence
- **Scope** - Finds problems that are more specific rather than general

Cognitive Walkthrough: Issues

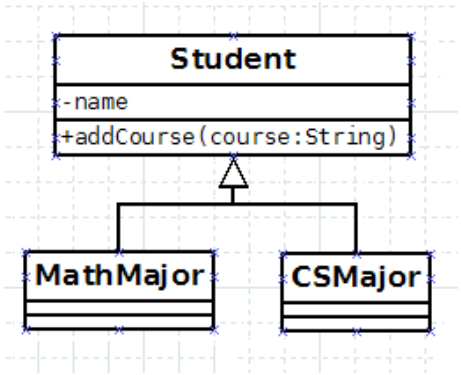
- Cannot evaluate every task the user will perform
  - Each user may have a different sequence of actions and control executions
- Each task is evaluated separately
  - The cross-task interactions are not identified
- A task-free user-centered method is brought in to catch problems that CW may have missed
  - Heuristic evaluation

Cognitive Walkthrough: Summary

- Assesses **learnability** (ease of learning) of a user interface
- Identifies **specific problems** with design
- No need for users to get involved

CW: A step-by-step Example

- Goal: Create UML diagram in DIA



Representative Task

Steps

1. Put in UML mode
2. Add parent class (Student)
   A. Select class tool
   B. Draw class onto canvas by clicking
   C. Change class name
3. Add name as private String
   A. Bring up dialog, click on Attribute tab
   B. Click New
   C. Enter name
   D. Change visibility to Private
   E. Click OK

Steps (cont.ed)

4. Add public method addCourse (String parameter)
   A. Click on Operations tab
   B. Press New
   C. Enter method name
   D. Click New parameter
   E. Enter parameter name (course)
   F. Enter parameter type (String)

Steps (cont.ed)

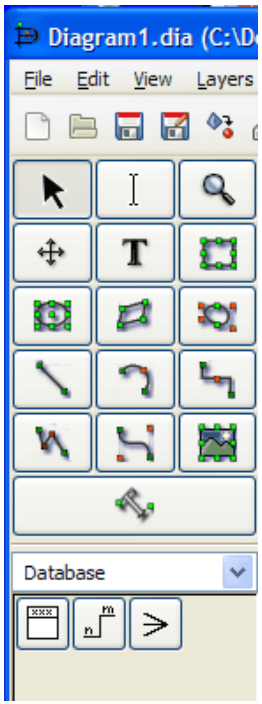
5. Add CSMajor and MathMajor as children
   A. Create CSMajor and MathMajor classes, as above
   B. Line them up on the canvas
   C. Select Generalization tool
   D. Drag mouse from parent class to one child
   E. Use Zigzagline to connect to second child

Step 4. Record Important Information

- User knowledge (just before and just after the action)
- Assumptions about users
- Side issues and design changes
- Credible success or failure story
  - i.e., why would a user select or not select the correct action?

Step 1: UML mode

- Screen comes up in database mode



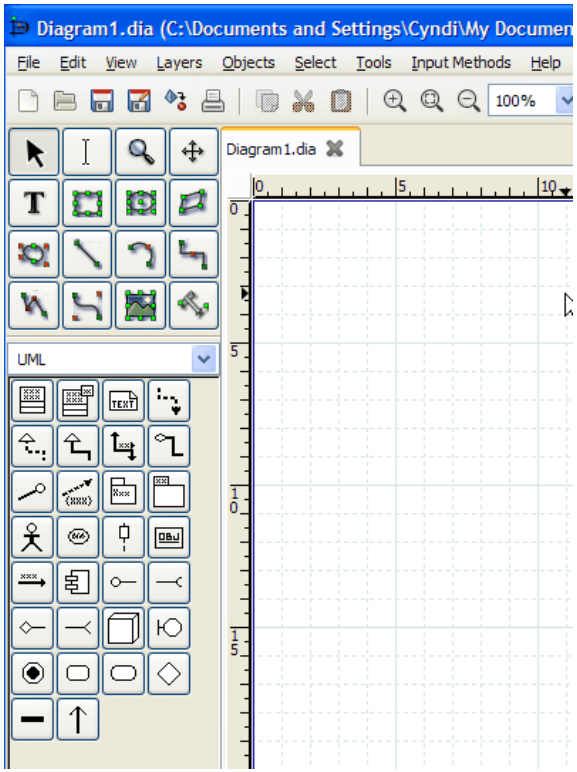
I'm thinking: I want to create a UML diagram

Action:
- I see a lot of symbols that aren't UML.
- I look through the menus, don't see UML.
- Finally notice drop down with Database. I try it. Now I see UML.

Recommendation:
- Highlight the drop down. It could be moved up, but those tools don't change. So it makes sense where it is, but it's mid-screen, hard to notice.
- Also, add a Diagram Type option to one of the menus, maybe Select.

Step 1: UML mode

- Now the UML menu is available.



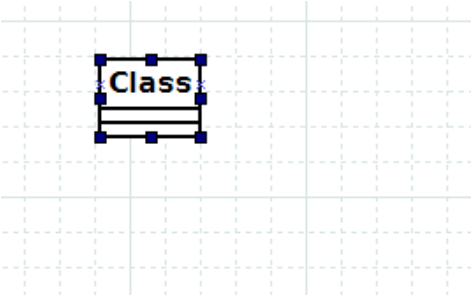
I'm thinking: I want to draw a class

Action:
- Scan the symbols. Tool tip for first one says class. I select it.

Recommend:
- Tool tips are effective. Class is first icon, seems reasonable. No issues with this step.

Step 2: Add parent class (Student)
Step 2A: Select class tool

- Now I've selected the class tool



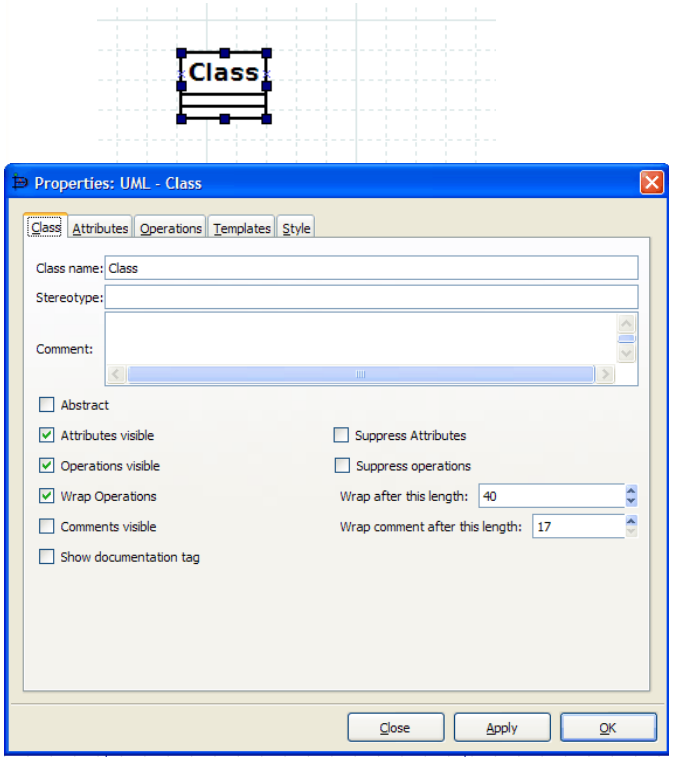
I'm thinking: OK, now I want to add Student to my diagram. Do I click or drag?

Action:
- I click on the canvas.
- Class is added, with name Class.

Recommend:
- This seems clear, no recommendation

Step 2: Add parent class (Student)
Step 2B: Draw class onto canvas by clicking

- Now I've added the class to my drawing



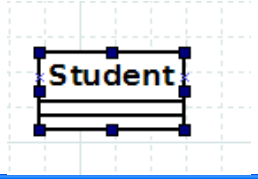
I'm thinking: I want to change the name to Student.

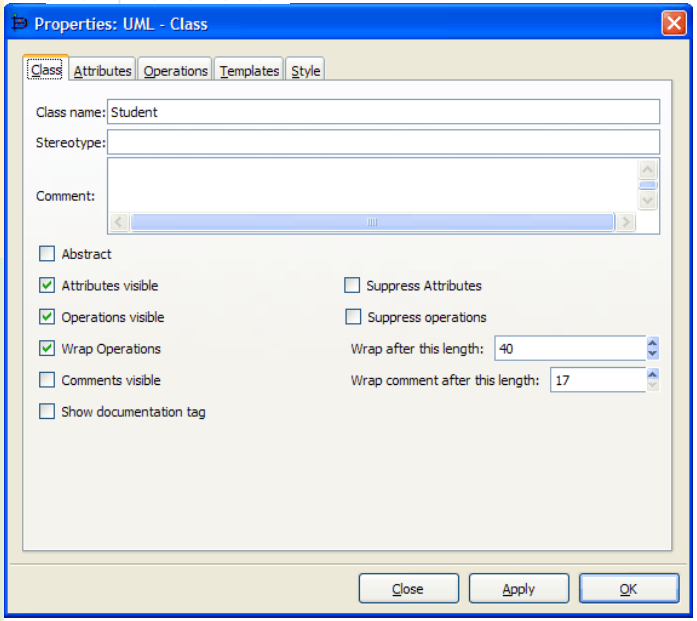
Action:
- I double-click where it says Class.
- Dialog comes up.
- First text field is Class name. I enter Student. Press OK.
- Class name is changed to Student.

Recommend:
- This seems clear, no recommendation

Step 3: Add name as private String
Steps 3A: Bring up dialog, click Attribute tab

- Now I've changed the class name to Student





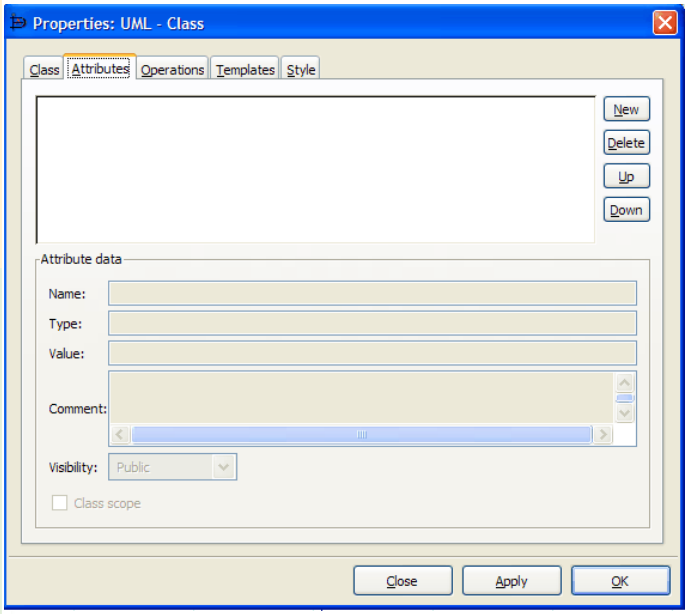
I'm thinking: OK, I want to add my fields. There were a lot of options on that dialog I just used.

Action:
- I double-click on Student class.
- Dialog appears.
- Checkboxes don't seem to apply. I notice Attributes (which I recognize as synonym for fields). Click on Attributes tab.

Recommend:
- Tabs probably OK for experienced users. Would a novice notice?
- Dialog has options I don't understand (e.g., Attributes visible vs. Suppress Attributes, Wrap options). Visual representation might be nice, if tool is for beginners.

Step 3: Add name as private String
Steps 3B: Click new

- Now I'm at the correct dialog



I'm thinking: I want to type in the variable info.

Action:

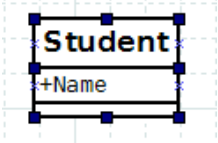
- I try to type in Name: field, but it's grayed out.
- I consider just typing into the big text box, but that doesn't seem right.
- I notice New, figure that's what I need.
- Click it, I'm able to enter a Name and Press OK.

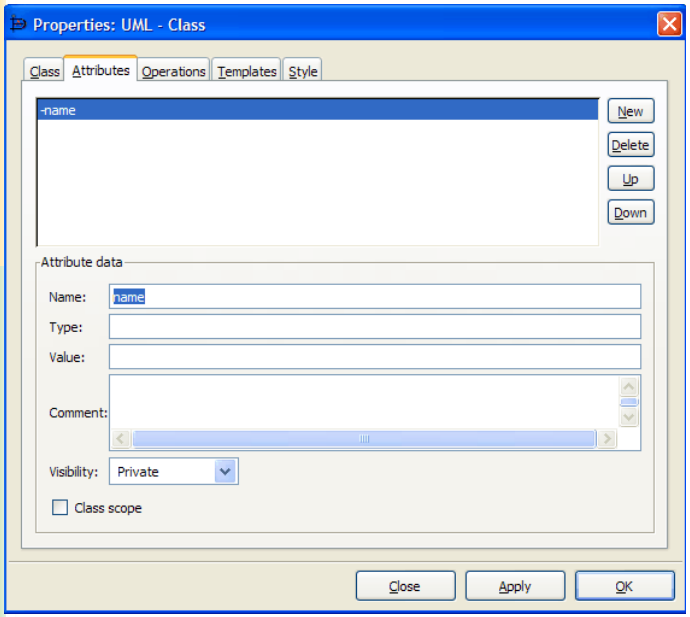
Recommend:

- We read right-to-left. I would probably put buttons on left side of text area. Maybe put default text such as "No attributes defined" in the text area.

Step 3: Add name as private String
Steps 3B: Change visibility to Private

- Now my attribute is listed, but it has a +





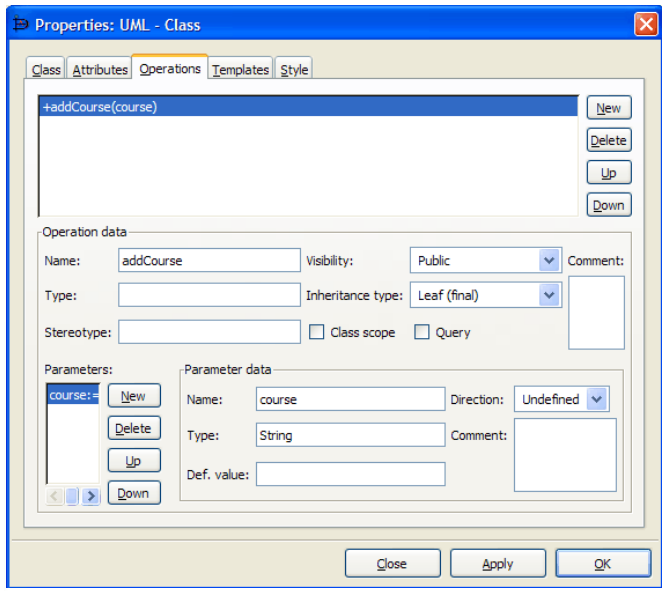
I'm thinking: I missed something.

Action:
- I bring dialog back up
- Click on name
- I quickly notice Visibility, change to Private

Recommend:
- I would default to Private (that's normally recommended except for constants)
- I would move Visibility higher in list, after Type or Value

Step 4: Add public method addCourse
Steps 4A – 4E

- Now my attribute is listed, but it has a +



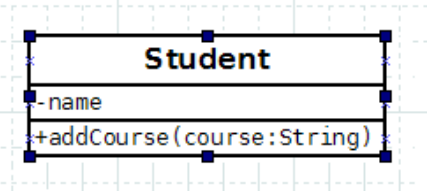
I want to add a method. I know now to look at the tabs. Methods is not there, but Operations is. Screen operation is similar to Attributes, so I immediately press New. I then enter the method Name. I press New under parameters. I enter the Name and Type.

Recommend:

- I would move buttons to left (as with suggestion for Attributes). Rest seems pretty intuitive.

Step 5: Add CSMajor and MathMajor as children
Step 5A:

- Now I have a fully defined parent class

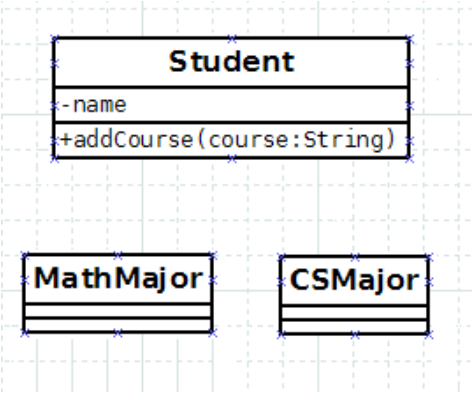


I'm thinking: I know how to create classes, first I need to create the two children.

Recommend:
- No recommendation

Step 5: Add CSMajor and MathMajor as children
Step 5B: Line them up on the canvas

- Now I have 3 classes

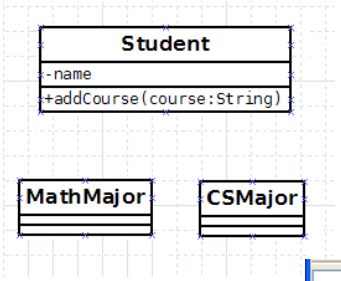


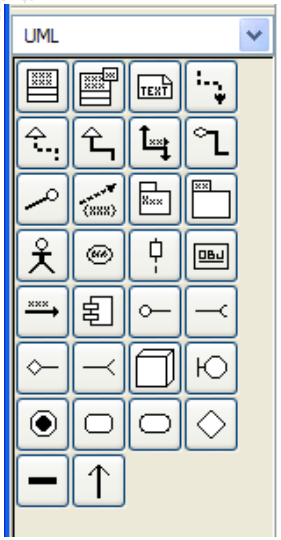
I'm thinking: The canvas is like most drawing programs, I can just click on the objects and move them.

Recommend:
- No recommendation

Step 5: Add CSMajor and MathMajor as children
Step 5C: Select Generalization tool

- Now I have 3 classes lined up





I'm thinking: I need to find the tool to draw an inheritance relationship.

Action:

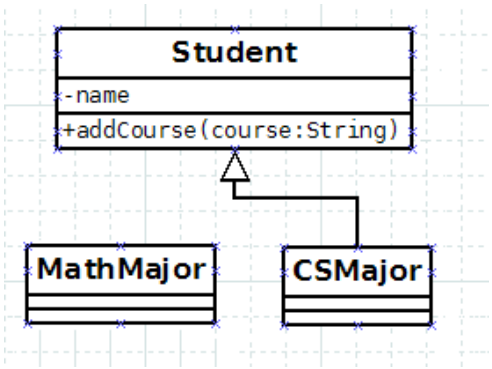
- Notice that the UML toolbar has a tool in the 2nd row that looks like generalization. Tool tip confirms.

Recommend:

- No recommendation

Step 5: Add CSMajor and MathMajor as children
Step 5D: Drag mouse from parent class to one child

- Now I have 3 classes and have selected inheritance tool



I'm thinking: This looks like a typical drawing tool. I should draw from the parent to the child.

Action:

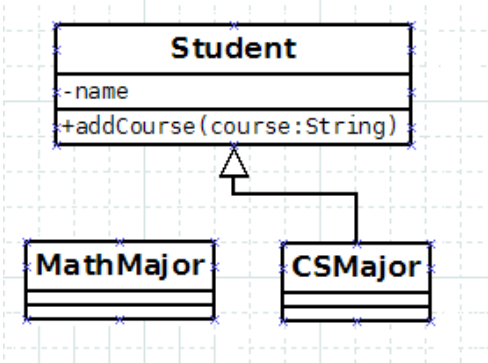
- Use tool to draw as expected. As I'm drawing I notice the connection points on the sides of the classes. Line snaps into place.

Recommend:

- No recommendation

Step 5: Add CSMajor and MathMajor as children
Step 5E: Use Zigzagline to connect to second child

- Now I have 3 classes and one inheritance relationship



I'm thinking: There should be an easy way to connect a second child.

Action:

- I try to click on existing line, but don't see any way to extend it to the 2nd class. I look at other tools at top of program. I notice the jagged line (tool tip says Zigzagline). Click on that, use to update drawing. *

Recommend:

- The drawing looks OK, but there doesn't seem to be any semantic meaning. It would be great to click on triangle, click on 2nd child, have the tool generate the line.

** There may be a better way to do this, but I haven't found it.*

CW: Example (2) – Google Forms

forms.google.com

- User: Average laptop/tablet user
- Analysts: YOU
- The task: Create a Google Form
  - Actions:
    - Make a new form
    - Give it a title
    - Add three questions: One selector, one short answer, one scale
    - Change the background
    - Share a link to the form

Study Material & Reading

- BOOK: Preece, J., Rogers, Y. and Sharp, H. Interaction Design.
  - Chapter: Evaluation: Inspections, Analytics & Models