

Human Computer Interaction

Eliminating Excuse and Scenarios

Prof. Andrew D. Bagdanov

Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Firenze
`andrew.bagdanov AT unifi.it`

December 1, 2017

- 1 News
- 2 Scenarios and Requirements
- 3 Eliminating Excise
- 4 Homework

News

Upcoming lectures:

- Tuesday, 5 December: Metaphors, idioms, and interaction.
- Wednesday, 6 December: Android Lab
- Friday, 8 December: NO LECTURE

Today:

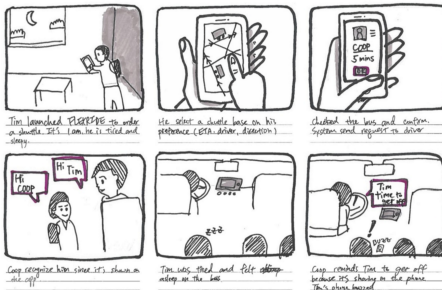
- Latest and greatest: Products as Agents.
- Leftovers: Final Words on Personas
- Lecture: Scenarios and Eliminating Excuse

Scenarios and Requirements

- Through careful analysis of **user research** and **synthesis of personas**, we create a clear picture of our users and their respective goals.
- We now come to the crux of the whole method: how to use this **understanding of people** to create design solutions that **satisfy and inspire users**.
- We must employ personas as the main characters in a set of techniques that arrive at design solutions in an iterative, repeatable, and testable fashion.
- This process has four major activities:
 - Developing **scenarios** as a means of **imagining ideal user interactions**.
 - Using those scenarios to **define requirements**.
 - Using these requirements in turn to define the **fundamental interaction framework** for the product.
 - And filling in the framework with increasing amounts of design **detail**

Scenarios: Narrative as a Design Tool

- Storytelling is one of the oldest human activities, and much has been written about the power of narrative to **communicate ideas**.
- Because interaction design deals with **behavior occurring over time**, a narrative structure is perfectly suited for representing and validating interaction concepts.
- Interaction design narratives are similar to the storyboards used in the motion picture industry: they have a **plot**, and they are **concise**.
- Just as storyboards breathe life into a movie script, design solutions should be **created and rendered to follow a plot**.



- In the 1990s, much work was done by the HCI community around the idea of **use-oriented software design**.
- From this work came the concept of the **scenario**: making use of a specific story to both construct and illustrate design solutions.
- John Carroll writes, in his book *Making Use*:

Scenarios are paradoxically concrete but rough, tangible but flexible ... they implicitly encourage “what-if?” thinking among all parties. They permit the articulation of design possibilities without undermining innovation ... Scenarios compel attention to the use that will be made of the design product. They can describe situations at many levels of detail, for many different purposes, helping to coordinate various aspects of the design project.

- **Persona-based scenarios** are concise narrative descriptions of one or more personas using a product to achieve specific goals.
- With them, we start our designs with an ideal experience, focusing on **people** rather than on **technology** or **business** goals.
- **Goals** serve as a filter for tasks and as **guides for structuring the display of information and controls** during the iterative process of constructing the scenarios.
- Designers **role-play personas** as the characters in these scenarios.
- This process leads to synthesis of structure and behavior and later informs the **detailed look-and-feel**.
- Personas and scenarios are also used to **test the validity** of design ideas and assumptions **throughout** the process.

The Context Scenario

- The **context scenario** is used to explore, at a high level, how the product can best serve the needs of the personas.
- These scenarios are created before any design and are written from the **perspective of the persona**, focused on human activities, perceptions, and desires.
- It is in the development of this kind of scenario that the designer has the most leverage to imagine an **ideal user experience**.

The Key Path Scenario

- After designing functional and data elements, a context scenario is revised to become a **key path scenario** by more specifically describing user interactions and introducing the **vocabulary of the design**.
- These scenarios focus attention on how a **persona uses the product to achieve their goals**.
- Key path scenarios are **iteratively refined** along with the design as more and more detail is developed.

- The **Requirements Definition** phase determines the **what** of the design: what information and capabilities our personas require to accomplish their goals.
- It is absolutely critical to define and agree upon the **what** before we move on to the next question: **how** the product looks, behaves, operates, and feels.
- Many designers are tempted to jump right into **active design** and render possible solutions.
- **Conflating these two questions is one of the biggest pitfalls in the design of an interactive product.**

- The **Requirements Definition** process for interaction design consists the following five steps:
 - 1 Creating problem and vision statements
 - 2 Brainstorming
 - 3 Identifying persona expectations
 - 4 Constructing context scenarios
 - 5 Identifying requirements
- Although these steps proceed in roughly chronological order, they are an **iterative** process.
- Expect to cycle through Steps 3 through 5 several times until the **requirements are stable**.

- It's important for designers to have a clear **mandate** for moving forward.
- At this point we already have a sense of which **users** we're targeting and what their goals are, but how do we know when to **design**?
- Problem and vision statements provide just such a mandate and help build consensus among stakeholders before the design process moves forward.
- A design **problem** statement should concisely reflect a situation that needs changing:

ACME's customer satisfaction ratings are low and market share has diminished by 10% over the past year because users don't have adequate tools to perform X, Y, and Z tasks that would help them meet their goal of G.

- The **vision statement** is an inversion of the problem statement that serves as a high-level design objective or mandate.
- In the vision statement, you lead with the **user's needs**, and you transition from those to how business goals are met by design vision.
- The content of both the problem and vision statements should come **directly from research and user models**.

The new design of Product X will help users achieve G by giving them the ability to perform X, Y, and Z with greater [accuracy, efficiency, and so on], and without problems A, B, C that they currently experience. This will dramatically improve ACME's customer satisfaction ratings and lead to increased market share.

Step 2: Brainstorming

- At the early stage of Requirements Definition, **brainstorming** assumes a somewhat **ironic** purpose.
- We have been researching and modeling users, so, it is almost impossible to avoid having developed some **preconceptions** about what the solution looks like.
- The reason we brainstorm at this point in the process is to get these ideas **out of our heads**.
- A side benefit of brainstorming is to switch our brains into **solution mode**.
- Much of the work performed in the Research and Modeling phases is **analytical in nature**, and it takes a different mindset to come up with inventive designs.

Step 3: Identifying persona expectations

- Mental models are deeply ingrained and are the result of a **lifetime of experience**.
- Expectations about a product and the way it works are **highly informed by their mental model**.
- It's critical that the model of the interface match the user's mental model rather than reflecting the **implementation model**.
- To accomplish this, for each primary and secondary persona we identify:
 - Attitudes, experiences, aspirations, and other social, cultural, environmental, and cognitive factors that **influence the persona's expectations**.
 - **General expectations** and desires the persona may have about the experience of using the product.
 - **Behaviors** the persona will expect or desire from the product.
 - How that persona thinks about **basic elements** or units of data.

Step 4: Constructing context scenarios

- Though all scenarios are stories about people and their activities, context scenarios are the most storylike of the three types we employ.
- The focus is on the persona's activities, as well motivations and mental model.
- Context scenarios describe the broad context in which usage patterns are exhibited.
- **This is where design begins:** as you develop context scenarios, you should be focusing on how the product you are designing can best help your personas achieve their goals.
- Context scenarios address questions such as:
 - In what setting(s) will the product be used?
 - Will it be used for extended amounts of time?
 - Is the persona frequently interrupted?
 - Are there multiple users on a single workstation or device?
 - With what other products will it be used?
 - What activities does the persona need to perform to meet goals?
 - What complexity is allowed, based on persona skill and frequency of use?

- The following is a **first iteration** of a context scenario for a primary persona for a personal digital assistant (PDA).
- Our persona is: Vivian Strong, a real-estate agent in Indianapolis, whose goals are to balance work and home life, close the deal, and make each client feel like he is her only client.
- Vivian's context scenario:
 - 1 While getting ready in the morning, Vivian uses her phone to check her e-mail. It has a large enough screen and quick connection time so that it's more convenient than booting up a computer as she rushes to make her daughter, Alice, a sandwich for school.
 - 2 Vivian sees an e-mail from her newest client, Frank, who wants to see a house this afternoon. The device has his contact info, so now she can call him with a simple action right from the e-mail.
 - 3 While on the phone with Frank, Vivian switches to speakerphone so she can look at the screen while talking. She looks at her appointments to see when she's free. When she creates a new appointment, the phone automatically makes it an appointment with Frank, because it knows with whom she is talking.

An example Context Scenario (continued)

- 4 After sending Alice off to school, Vivian heads into the real-estate office to gather some papers for another appointment. Her phone has already updated her Outlook appointments, so the rest of the office knows where she'll be in the afternoon.
- 5 The day goes by quickly, and she's running a bit late. As she heads towards the property she'll be showing Frank, the phone alerts her that her appointment is in 15 minutes. When she flips open the phone, it shows not only the appointment, but a list of all documents related to Frank, including e-mails, memos, phone messages, and call logs to Frank's number. Vivian presses the call button, and the phone automatically connects to Frank because it knows her appointment with him is soon. She lets him know she'll be there in 20 minutes.
- 6 Vivian knows the address of the property but is a bit unsure exactly where it is. She pulls over and taps the address she put into the appointment. The phone downloads directions along with a thumbnail map showing her location relative to the destination.

- 7 Vivian gets to the property on time and starts showing it to Frank. She hears the phone ring from her purse. Normally while she is in an appointment, the phone will automatically transfer directly to voicemail, but Alice has a code she can press to get through. The phone knows it's Alice calling, and uses a distinctive ring tone.
- 8 Vivian takes the call – Alice missed the bus and needs a pickup. Vivian calls her husband to see if he can do it. She gets his voicemail; he must be out of service range. She tells him she's with a client and asks if he can get Alice. Five minutes later the phone makes a brief tone Vivian recognizes as her husband's; she sees he's sent her an instant message: "I'll get Alice; good luck on the deal!"

Step 5: Identifying Requirements

- When satisfied with an initial draft of a context scenario, you can analyze it to extract persona **needs or requirements**.
- These requirements can be thought of as consisting of **objects**, **actions**, and **contexts**.
- A need from the scenario above might be:

Call (action) a person (object) directly from an appointment (context).

- It can also be helpful to separate needs into **data** and **functional** requirements, as described in the following.

Step 5: Types of Requirements

Data requirements

- Persona **data needs** are the objects and information that must be represented in the system.
- It is often useful to think of data requirements as the **objects and adjectives related to those objects**.
- Common examples: accounts, people, documents, messages, songs, images, as well as attributes of those such as status, dates, size, creator, subject, and so on.

Functional requirements

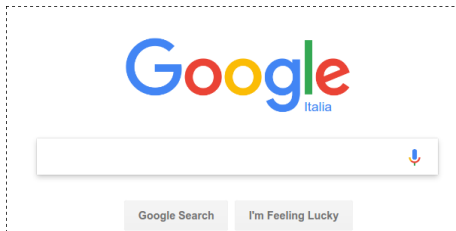
- Functional needs are the **operations or actions** that need to be performed on the objects of the system and which are typically translated into interface controls.
- These can be thought of as the **actions** of the product.
- Functional needs also define places or **containers** where objects or information in the interface must be **displayed**.

- It's important to get a firm idea of the realistic requirements of the business and technology you are designing for.
- Other requirements can include:
 - **Business requirements** can include development timelines, regulations, pricing structures, and business models.
 - **Brand and experience requirements** reflect attributes of the experience you would like users and customers to associate with your product, company, or organization.
 - **Technical requirements** can include weight, size, form factor, display, power constraints, and software platform choices.
 - **Customer and partner requirements** can include ease of installation, maintenance, configuration, support costs, and licensing agreements.

Eliminating Excise

- Software too often contains interactions that are **top-heavy**, requiring extra work for users.
- The result is software that charges its users a tax, or **excise**, of cognitive and physical effort every time it is used.
- A large task like driving to work involves many **smaller** tasks.
- Some of these tasks work **directly** towards achieving the goal, like steering down the road towards your office.
- **Excise tasks**, on the other hand, don't contribute directly to reaching the goal, but are **necessary to accomplishing it** just the same.
- Such tasks include finding your car, starting the engine, and stopping at traffic lights, in addition to putting oil and gas in the car and performing periodic maintenance.

- **Excise** is the extra work that satisfies either the needs of our **tools** or those of **outside agents** as we try to achieve our objectives.
- The distinction is sometimes hard to see because we get so **used to the excise being part of our tasks**.
- The problem with excise tasks is that the effort we expend in doing them **doesn't go directly towards accomplishing our goals**.
- By eliminating excise, we make people more **effective** and **productive**, and ultimately create a **better user experience**.



- One of the main criticisms of graphical user interfaces by users accustomed to command-line systems is that users must expend **extra effort manipulating windows** and menus to accomplish something.
- With a command line, users can just **type** in a command and the computer executes it immediately.
- With windowing systems, they must open various folders, after launching an application they must stretch and drag the window until it is in the desired location and configuration.
- These complaints are well founded: extra window manipulation tasks are, indeed, **excise**.
- But everybody knows that GUIs are easier to use than command-line systems. **Who is right?**

- The confusion arises because the real issues are hidden: the command-line interface forces an even more expensive excise budget on users because they must first **memorize** the commands.
- The excise of the command-line interface becomes smaller only after a user has invested **time and effort** in learning it.
- So, what are the principal forms of **GUI Excise**? And how can we **eliminate excise**?

- Expert users will learn each **nuance** of the applications they use.
- They will start up each application with a **clear idea** of exactly **what** they want to do and **how** they want to do it.
- To this user, the assistance offered to the casual or first-time user is **just in the way**.
- But we must be careful when we eliminate excise: we must not remove it just to suit **power users**.
- Similarly, however, we must not force power users to pay the full price for providing help to **new** or **infrequent users**.

- We can inadvertently introduce **significant excise** in support for first-time or casual users.
- It is **easy** to justify adding facilities to a product that will make it easy for newer users to learn how to use it.
- Unfortunately, these facilities quickly become **excise** as users become familiar with the application.
- Facilities added to software for the purpose of training beginners, such as step-by-step wizards, must be **easily turned off**.
- Training wheels are rarely needed for extended periods of time and are a **hindrance** to advanced users when they are left on permanently.

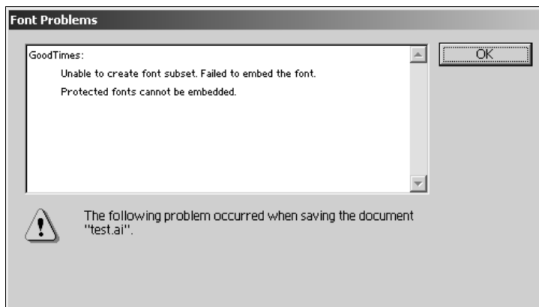
- **Visual excise** is the work that a user has to do to decode visual information.
- This includes finding a single item in a list, figuring out where to begin reading, or determining which elements are **clickable** and which are **merely decoration** (or **worse**).
- A significant source of visual excise is the use of **excessively stylized graphics** and interface elements.
- The use of visual style should **always** in support of the communication of information and interface behavior.
- Excessive ornamentation detracts from user effectiveness by forcing them to **decode visual elements** to understand which are **controls**.

GUI Excise: Visual Excise



- Certain tasks are **mainly** excise but can be useful for occasional users or users with special preferences.
- In this case, consider the function excise **if it is forced on a user** rather than made available **at his discretion**.
- The only way to determine whether a function or behavior such as this is excise is by **comparing it to persona goals**.
- If a primary persona needs to see **two applications** at a time on the screen, the ability to **configure the main windows** so they share the screen space is **not** excise.
- If your personas don't have this specific goal, the work required to configure the main window of either application **is excise**.

- One form of excise is so prevalent that it deserves special attention.
- Flow is a natural state, and it takes some effort to interrupt flow after someone has achieved it.
- Interruptions like a ringing telephone will do it, as will an **error message box**.
- Interrupting flow for no good reason is **stopping the proceedings with idiocy** and is one of the most disruptive forms of excise.

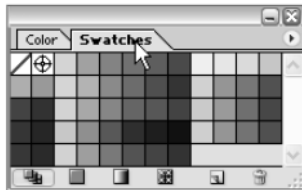
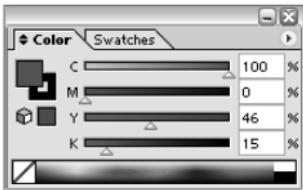


- This list will help us spot excise transgressions
 - Don't force users to **go to another window** to perform a function that affects the current window.
 - Don't force users to **remember where they put things** in the hierarchical file system.
 - Don't force users to **resize windows unnecessarily** –when a child window pops up on the screen it should be sized appropriately.
 - Don't force users to **move windows**. If there is open space on the desktop, put the application there instead of directly over some other already open program.
 - Don't force users to **reenter personal settings**. If a person has ever set a font, a color, an indentation, or a sound, make sure she doesn't **have** to do it again.
 - Don't force users to fill fields to satisfy some **arbitrary measure of completeness**.
 - Don't ask users to **confirm their actions** (this requires a robust undo facility).
 - Don't let a user's actions **result in an error**.

- The most important thing to realize about navigation is that it is largely **excise**.
- The work that users are forced to do to get around in software is **seldom aligned with their needs**, goals, and desires.
- **Unnecessary or difficult navigation is a major frustration to users**
- Navigation through software occurs at multiple levels:
 - Among multiple windows, views, or pages
 - Among panes or frames within a window, view, or page
 - Among tools, commands, or menus
 - Within information displayed in a pane or frame (for example: scrolling, panning, zooming, following links).
- Navigation is: **any action that takes a user to a new part of the interface or which requires him to locate objects, tools, or data.**

- Navigation among multiple application views or Web pages is perhaps the most **disorienting** kind of navigation for users.
- It involves **shifting of attention** and **flow disruption**.
- If users must constantly shift back and forth between windows to achieve their goals, their **disorientation** and **frustration** levels will rise.
- If the number of windows is large enough, a user will become sufficiently disoriented that he may experience navigational trauma and **get lost in the interface**.
- **Sovereign posture applications** can avoid this problem by placing all main interactions in a **single primary view**, which may contain **multiple independent panes**.

- **Tabbed panes** can be appropriate when there are multiple supporting panes that are not used at the same time.
- The support panes can then be **stacked**, and a user can choose the pane suitable for his current tasks, which is only a single click away.
- A classic example here is the color mixer and swatches area in Adobe Illustrator.



- There are many ways improve navigation in our applications, Web sites, and devices.
- Here are some:
 - Reduce the **number** of places to go.
 - Provide **signposts**.
 - Provide **overviews**.
 - Provide appropriate **mapping** of controls to functions.
 - Inflect your interface to **match user needs**.
 - Avoid **hierarchies**.

- The most effective method of improving navigation sounds quite obvious: **reduce the number of places to which one must navigate.**
- These “places” include modes, forms, dialogs, pages, windows, and screens.
- This means:
 - **Keep the number of windows and views to a minimum.** One full-screen window with two or three views is best for many users.
 - **Keep the number of adjacent panes in your window or Web page limited** to the minimum number needed for users to achieve their goals. Three panes is a good target, but there are no absolutes.
 - **Keep the number of controls limited** to as few as your users really need to meet their goals. Having a good grasp of your users via personas will enable you to avoid functions and controls that your users don't really want or need.
 - **Scrolling should be minimized when possible.** This means giving supporting panes enough room to display information so that they don't require constant scrolling.

- Another way to enhance user ability to find their way around is by providing better points of reference, or **signposts**.
- Each application most likely has a main, top-level window.
- The salient features of that window should be considered **persistent objects**: menu bars, toolbars, and other palettes or visual features like status bars and rulers.
- Well-designed Web sites make careful use of **persistent objects** that remain constant.
- These provide clear **navigational options**, and their consistent presence and layout also help orient users.

GUI Excise: Provide Signposts

Amazon.it: lenovo yoga

www.amazon.it/s/ref=se_ex_n_0?rh=i%3Aaaps%2Ck%3Alenovo+yoga&keywords=lenovo+yoga&ie=UTF-8

amazon.it

Scegli per categoria - Amazon.it di Andy Offerte Buoni Regalo Vendere Aiuto

Giao Andy Il mio account - Iscriviti a Prime - Le tue Liste -

Amazon.it Offerte Usato e ricondizionato Outlet Made in Italy Novità Bestseller Amazon Prime App di Amazon Lista Desideri Buoni regalo Vendi usato

1-16 dei 57.685 risultati in "lenovo yoga" Ordina per

Mostra risultati in

Informatica >
Tablet PC
Portatili
* Vedi altri


* Tutte le 18 categorie

Filtra per

Modalità di spedizione (Cov'9)
☒ Prime
☐ Spedizione gratuita


Marca
☐ Lenovo
☐ Asus
☐ Emarbuy®
☐ Tuff-Luv
☐ MoKo
☐ Lavolta
☐ Displayschutz@Folix
☐ IBM / Lenovo
☐ Memori Memory
☐ 2-tech
☐ N/A
☐ Evecase
☐ Navitech

Ricerca correlata lenovo yoga 3 pro, lenovo yoga 500.


Lenovo YOGA 500-14IBD, Notebook con Windows 10 Home, Intel Core Processor i3-5005U, RAM 4GB, HDD 500GB, Display da 14.0" HD 1366x768, Layout Italiano, Bianco
di Lenovo
EUR 540,34 ~~EUR 599,00~~ ☒ Prime
Ricevilo entro **venerdì 20 maggio**
Ulteriori opzioni di acquisto
EUR 533,36 nuovo (40 offerte)


★★★★☆ 1

Descrizione prodotto
... LENOVO YOGA 500-14IBD 14" TOUCH SCREEN i3 2GHZ RAM 4GB-HDD 500GB-GEFORCE ...
Elettronica: Visualizza tutti e 22.087 gli articoli


Lenovo Yoga 300-11BY Notebook, Windows 8.1, Intel Celeron N2840, RAM 2GB, HDD 32GB, Display da 11.6" HD 1366x768, Layout Italiano, Bianco
di Lenovo
~~EUR 329,00~~ **EUR 303,50** + EUR 8,00 di spedizione
Solo 3 con disponibilità immediata - Ordina ora.
Ulteriori opzioni di acquisto
EUR 303,50 nuovo (9 offerte)

★★★★☆ 7

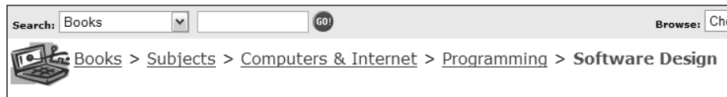
Descrizione prodotto
... Il design esclusivo del Lenovo Yoga 300 da 11" consente una rotazione ...
Elettronica: Visualizza tutti e 22.087 gli articoli


Lenovo Yoga 3 Pro Ultrabook Convertibile Multimodalità, 13.3", Processore Intel Core M-5Y70, 8 GB RAM, SSD da 512 GB, Oro
di Lenovo
EUR 954,52 ~~EUR 1.599,00~~ ☒ Prime
Solo 1 con disponibilità immediata - Ordina ora.

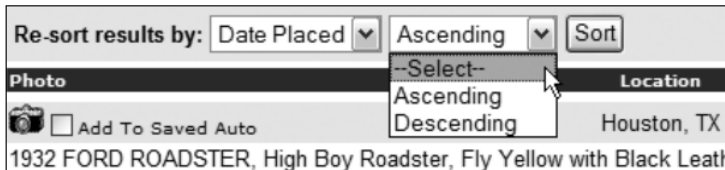
★★★★☆ 8

Descrizione prodotto

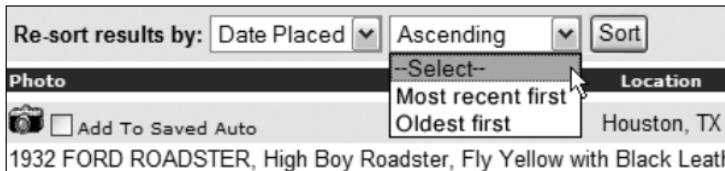
- Overviews serve a similar purpose as signposts: they help to **orient** users.
- The difference is that overviews help orient users within the **content** rather than within the application as a whole.
- Because of this, the overview area should itself be persistent: its content is **dependent on the data** being navigated.



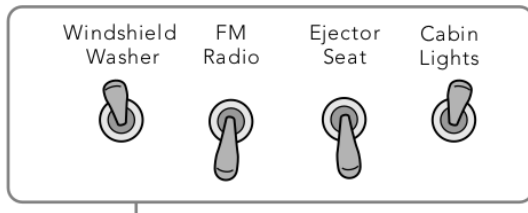
- Mapping describes the relationship between a control, the thing it affects, and the **intended** result.
- Poor mapping is when a control does not relate **visually or symbolically** with the object it affects.
- Poor mapping requires users to **stop and think about the relationship**, breaking flow.



- Mapping describes the relationship between a control, the thing it affects, and the **intended** result.
- Poor mapping is when a control does not relate **visually or symbolically** with the object it affects.
- Poor mapping requires users to **stop and think about the relationship**, breaking flow.



- **Inflecting** an interface is organizing it to minimize typical navigation.
- Controls and displays should be organized in an interface by:
 - **Frequency of use**: tools are most frequently used (many times a day) should be immediately in reach. Others (used perhaps once or twice a day) should be a click or two away.
 - **Degree of dislocation**: is the amount of sudden change in an interface caused by a specific function or command. Put these deeper into the interface.
 - **Degree of risk exposure**: deals with functions that are irreversible or dangerous. You should make these types of functions more difficult for your users to stumble across.



- Hierarchies are one of the programmer's most durable tools.
- Much of the data inside applications (along the code that manipulates it), is in hierarchical form.
- For this reason, many programmers present hierarchies (the implementation model) in user interfaces.
- But abstract hierarchies are very difficult for users to successfully navigate, except where they're based on user mental models.
- Most mechanical storage systems are simple, composed either of a single sequence of stored objects (like a bookshelf) or a series of sequences, one level deep.

Homework

Exercise 21b.1: Think about tech triggers.

What are the technology triggers of today that will give rise to the next wave of hype? How will this hype affect us in 10 years? How will it change **your** career over the next ten years? In twenty years?