Information Architecture

GUI Design

Graphical User Interface (GUI) Design focuses on anticipating what users might need to do and ensuring that the interface has elements that are easy to access, understand, and use to facilitate those actions.

UI brings together concepts from **interaction design, visual design, and information architecture.**

Interface structures

An organizational structure is how you define the relationships between pieces of content.

Successful structures allow users to predict where they will find information on the site.

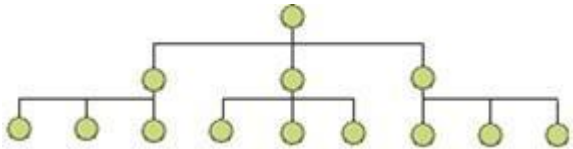
It’s important to take into account user expectations and implement consistent methods of organizing and displaying information so that users can extend their knowledge from familiar pages to unfamiliar ones.

The four main organizational structures are **Hierarchical, Sequential, Matrix and database model.**

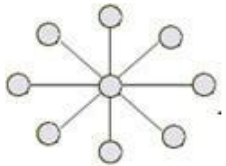
source: https://www.usability.gov/how-to-and-tools/methods/organization-structures.html

Hierarchical Structures

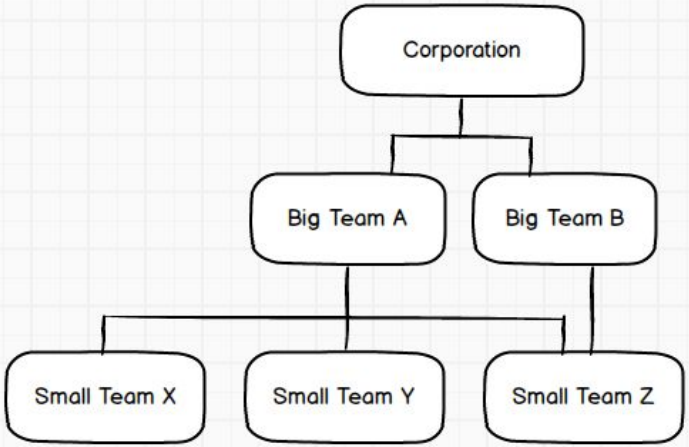
In Hierarchical Structures, sometimes referred to as tree structures or hub-and-spoke structures, there is a top down approach or parent/child relationships between pieces of information.



Users start with broader categories of information (parent) and then drill further down into the structure to find narrower, more detailed information (child).

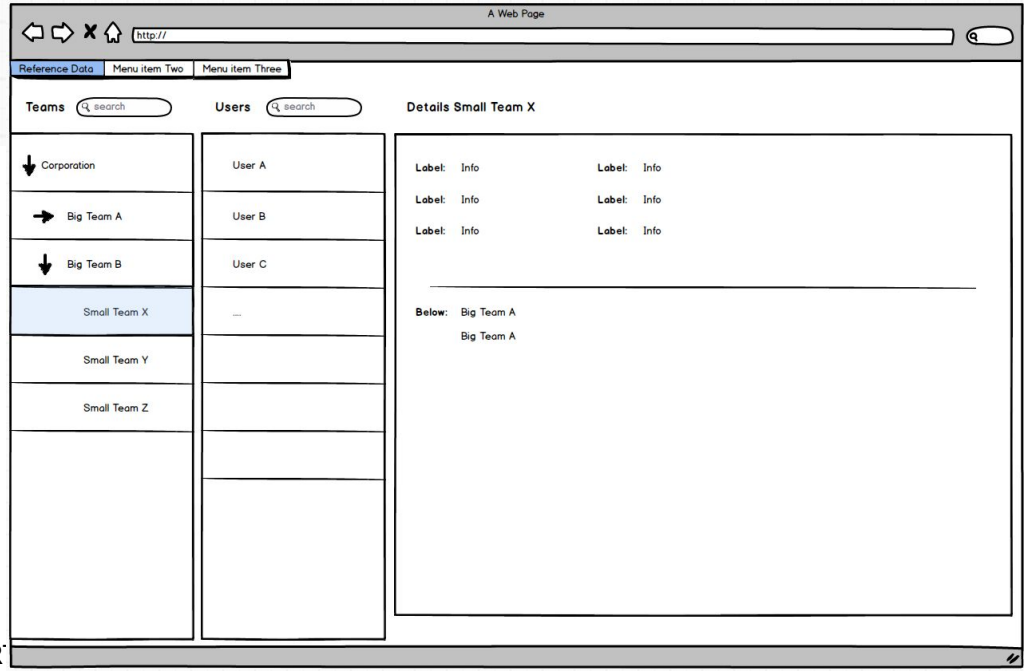


Many users are familiar with structuring information in hierarchies because they see these structures on a daily basis in the way businesses have a formed their lead leadership structure, the way project plans are set-up, and so on.



User

Account

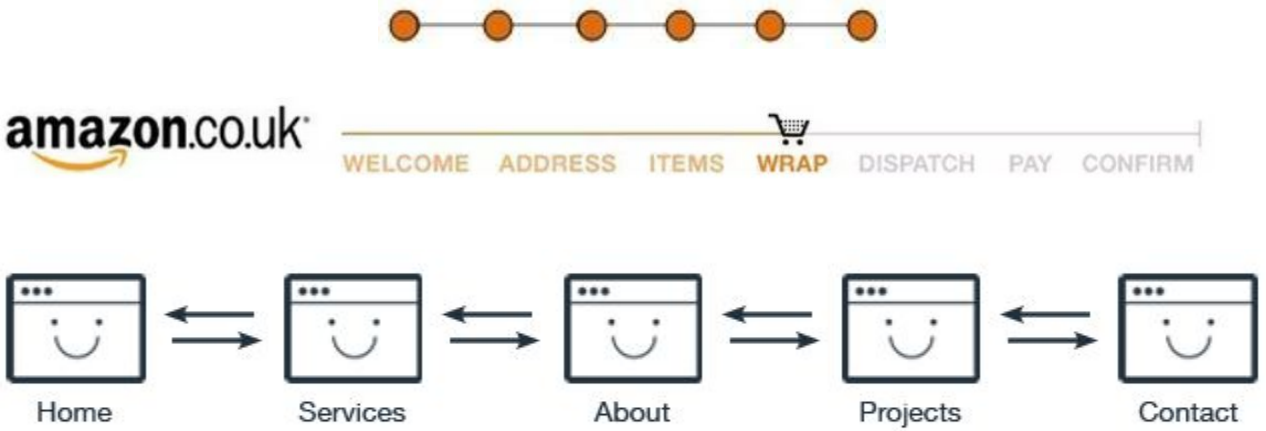


https://ux.stackexchange.com/questions/90872/display-hierarchical-structure-and-corresponding-reference-data

Sequential Structures

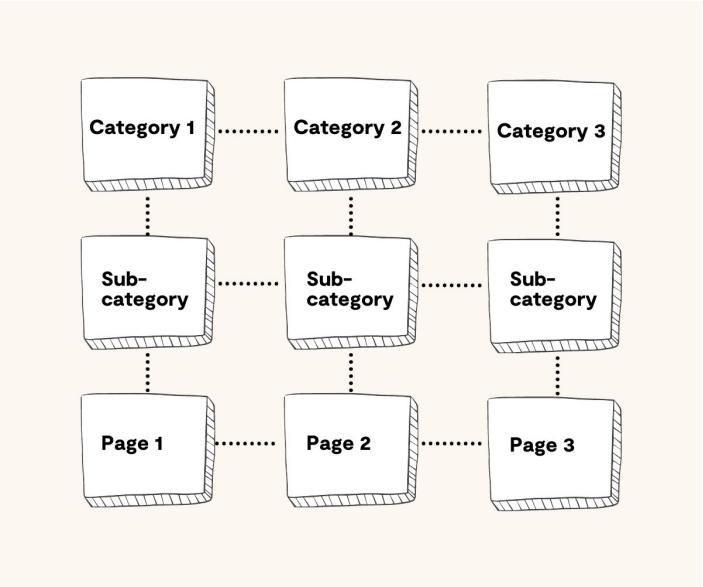
Websites with Sequential Structures require users to go step-by-step, following a specific path through content.

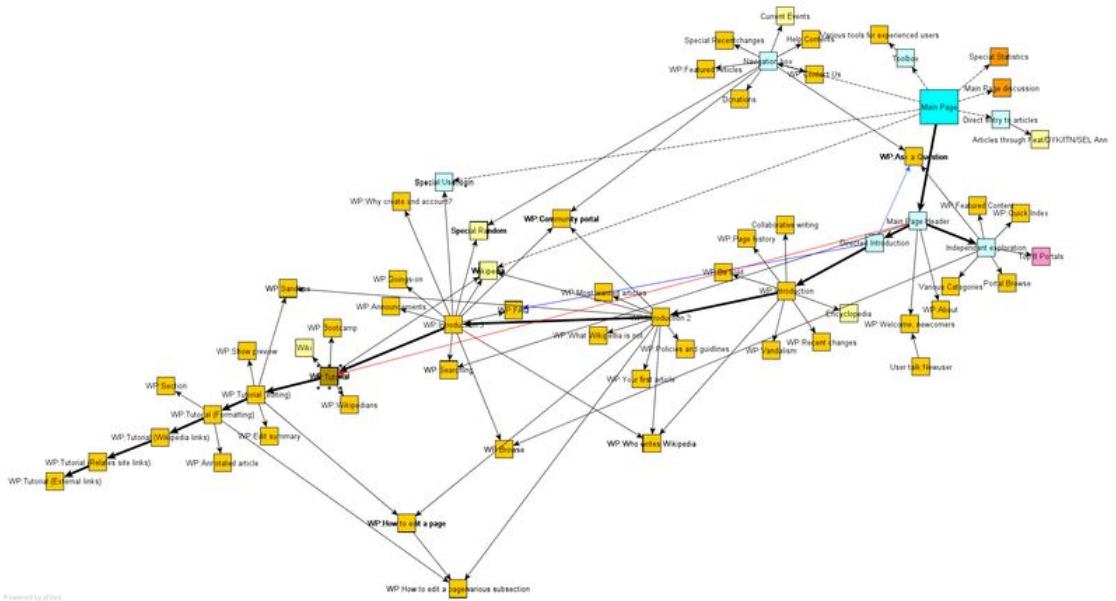
An example of this type of structure is when a user is attempting to purchase something or are taking a course online. Sequential structures assume that there is some optimal ordering of content that is associated with greater effectiveness or success.



Matrix Structures

A Matrix Structure allows users to determine their own path since content is linked in numerous ways. This type of structure takes full advantage of the principles behind hypertext, or HTML. For example, one user could choose to navigate through a set of content based on date while another navigates based on topic.





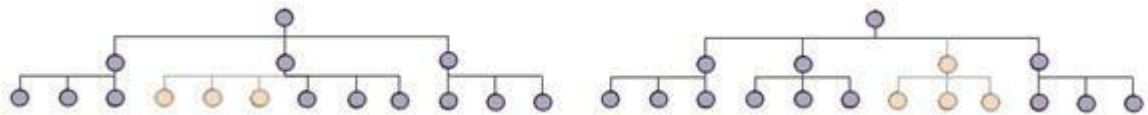
Database Model

The Database Model takes a bottom-up approach. The content within this structure leans heavily on the linkages created through the content's metadata. This type of model facilitates a more dynamic experience generally allowing for advanced filtering and search capabilities as well as providing links to related information in the system that has been properly tagged.

Creating Sustainable Structures

Site/GUI architecture has a long term impact on the site. It's important to put thought into the structure and ensure that it takes into account content updates in the future. Site managers should keep in mind the following when structuring a site:

Allow room for growth. Creating a site that can accommodate the addition of new content within a section (left image) as well as entire new sections (right image).

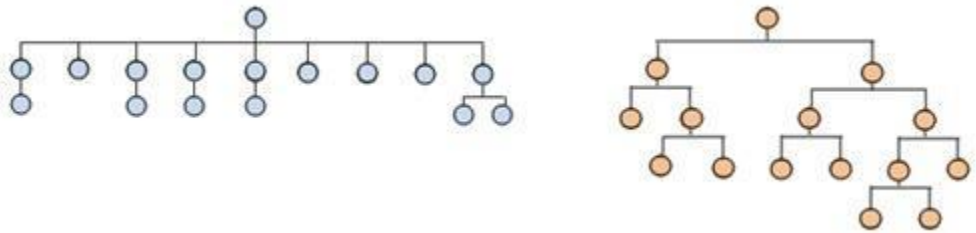


Creating Sustainable Structures

Avoid structures that are too shallow or too deep.

Striking a balance is never easy is an important goal of any architecture. Structures that are too shallow require massive menus.

Users rely on information architects to create logical groupings to facilitate movement throughout the site. In contrast, structures that are too deep bury information beneath too many layers. These structures burden the user to have to navigate through several levels to find the content that they desire.



Information Architecture

Information Architecture

Information architecture (IA) focuses on organizing, structuring, and labeling content in an effective and sustainable way. The goal is to **help users find information** and complete tasks.

To do this, you need to understand how the pieces fit together to create the larger picture, how items relate to each other within the system.

Why a Well Thought Out IA Matters?

According to Peter Morville Site exit disclaimer, the **purpose of your IA is to help users understand where they are, what they've found, what's around, and what to expect.** As a result, your **IA informs the content strategy** through identifying word choice as well as informing user interface design and interaction design through playing a role in the wireframing and prototyping processes.

Information Architecture

To be successful, you need a diverse understanding of industry standards for creating, storing, accessing and presenting information. Lou Rosenfeld and Peter Morville in their book, Information Architecture for the World Wide Web, note that the main components of IA are:

- **Organization Schemes and Structures:** How you categorize and structure information
- **Labeling Systems:** How you represent information
- **Navigation Systems:** How users browse or move through information
- **Search Systems:** How users look for information

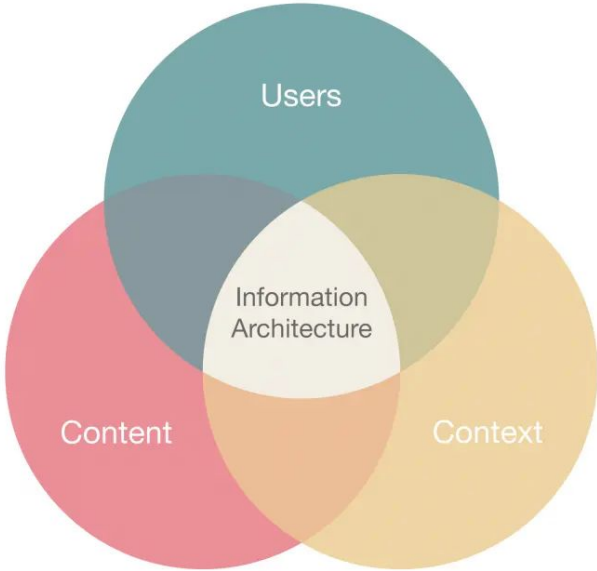
Information Architecture

In order to create these systems of information, you need to understand the interdependent nature of users, content, and context.

Rosenfeld and Morville referred to this as the “information ecology” and visualized it as a venn diagram.

Each circle refers to:

- **Context:** business goals, funding, politics, culture, technology, resources, constraints
- **Content:** content objectives, document and data types, volume, existing structure, governance and ownership
- **Users:** audience, tasks, needs, information-seeking experience



Organization Schemes

Organization schemes have to do with how you are going to categorize your content and the various ways you'll create relationships between each piece. Most content can be categorized in multiple ways.

Schemes can be broken down into **Exact and Subjective.**

Depending on the content, it’s conceivable that the site may combine schemes as opposed to treating them independently.

Exact Organization Schemes

Exact organization schemes objectively **divide information into mutually exclusive sections**.

These systems comparatively are easy for information architects to create and categorize content within. However, they can be a challenge at times for users.

It requires that the user understands how what they are looking for fits within the model.

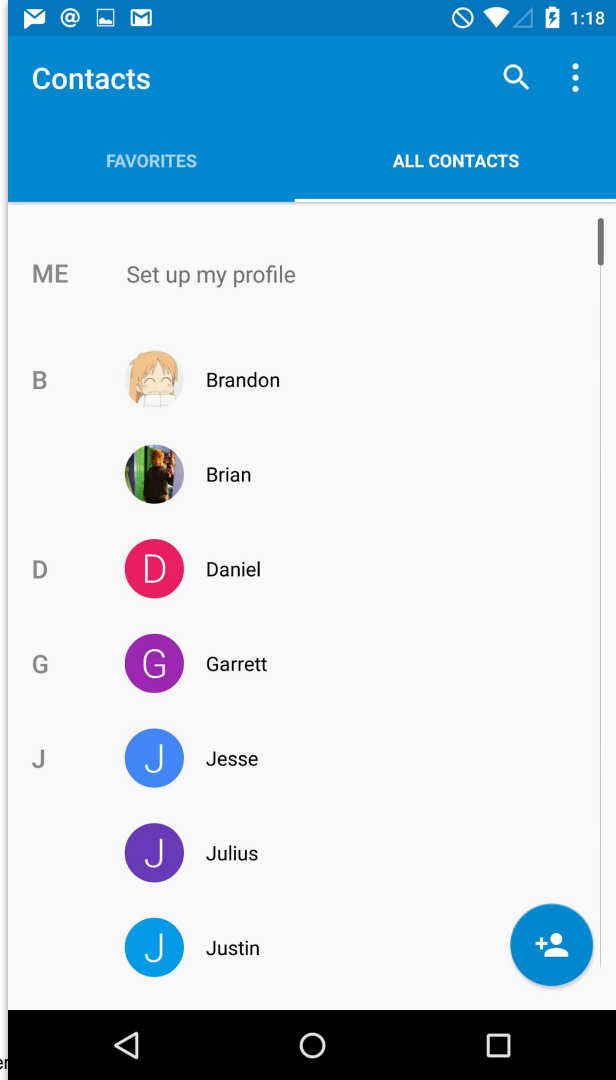
Examples of exact organizational structures include:

Exact Organization Schemes

Alphabetical schemes make use of our 26-letter alphabet for organizing their contents.

For this type of scheme to be successful, it is important that the content labeling matches the words that users are looking for.

Sometimes, alphabetical schemes in the form of an A-Z index serve as secondary navigational components to supplement content findability that is otherwise organized.

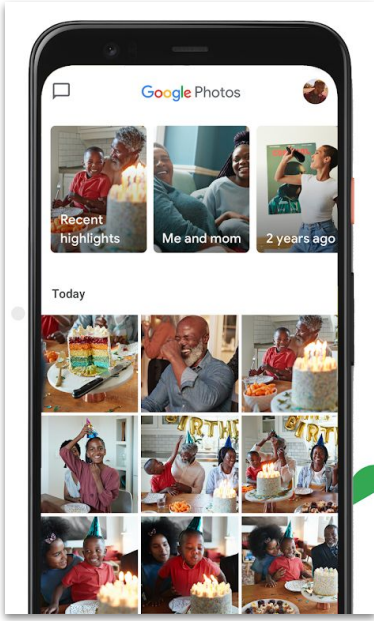


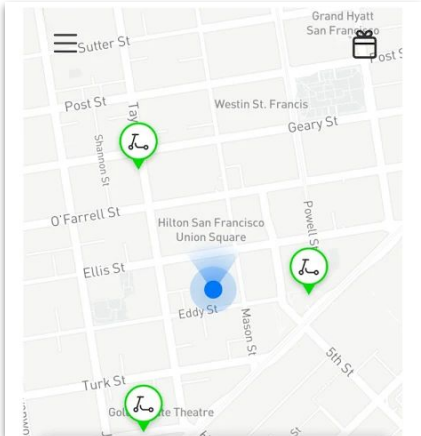
Exact Organization Schemes

Chronological schemes organize content by date. For these schemes to be successful there must be agreement about when the subject of the content took place.

Geographical schemes organize content based on place. Unless there are border disputes, this type of scheme is fairly straightforward to design and use.

Often these types of schemes serve as a good supplemental way to navigate a site that that is otherwise organized. For example, you may choose to provide a map to display information or an A-Z index to get to topics grouped primarily by one of the following subjective schemes.





Subjective Organization Schemes

Subjective organization schemes **categorize information in a way that may be specific** to or defined by the organization or field.

Although they are difficult to design, **they are often more useful than exact organization schemes.**

When information architects take the time to consider the user's mental models and group the content in meaningful ways, these types of schemes can be quite effective in producing conversions.

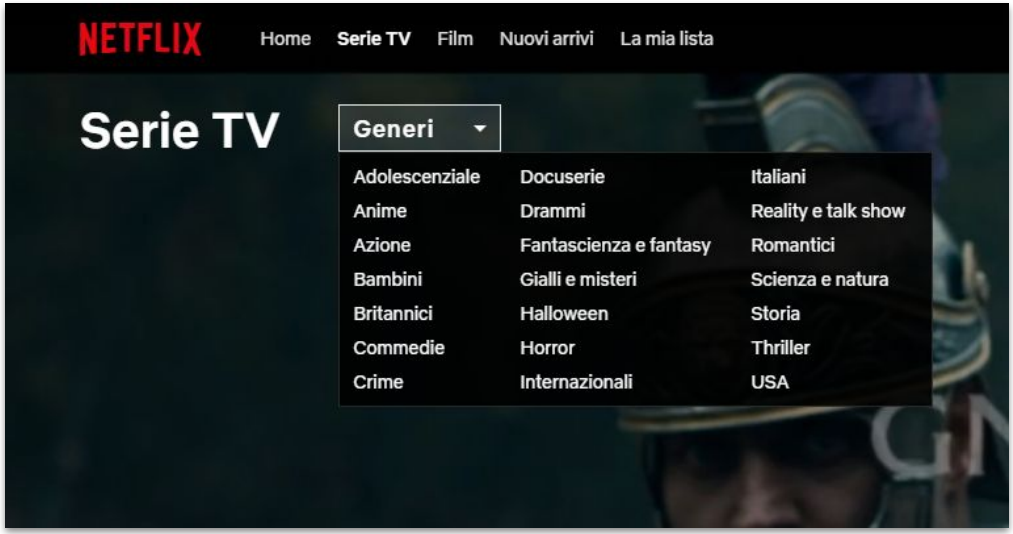
This type of categorization can also help facilitate learning by helping users understand and draw connections between pieces of content.

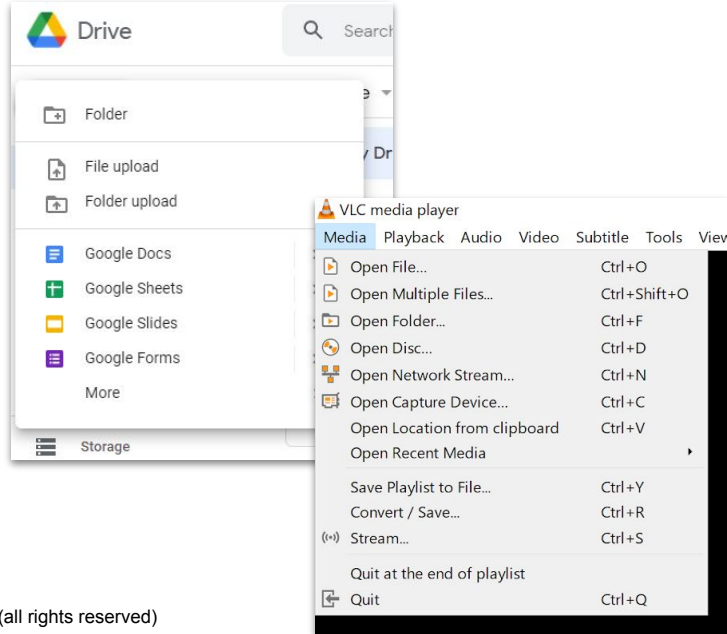
Subjective Organization Schemes

Examples of subjective schemes include the following:

Topic schemes organize content based on the specific subject matter.

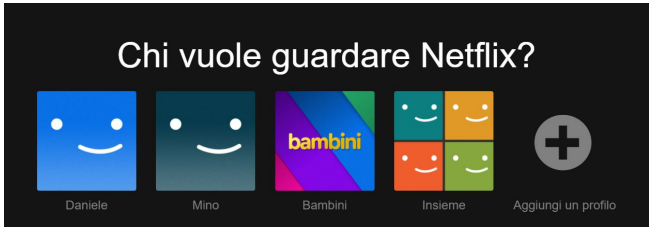
Task schemes organize content by considering the needs, actions, questions, or processes that users bring to that specific content.



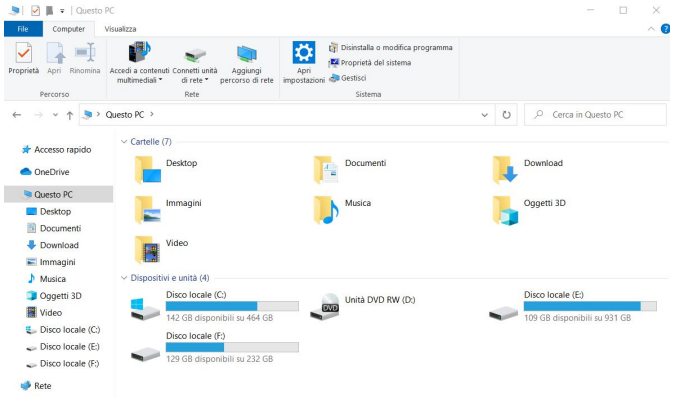


Subjective Organization Schemes

Audience schemes organize content for separate segments of users. Audience schemes can be closed or open, meaning that users are able to navigate from one audience to another. This type of scheme does present challenges unless the content lends itself to users very easily self-identifying to which audience they belong and perhaps not fitting multiple audience profiles.



Metaphor schemes help users by relating content to familiar concepts. This is used in interface design (folders, trash, etc) but can pose challenges when used as the site's primary organization scheme.



Challenges of Creating Hybrids

Implementing schemes independently has its advantages because it keeps things simple for the user.

They can identify the categorization and form a mental model that can be quickly understood.

Mixing schemes by creating hybrids can cause confusion for users.

This is often proposed as a solution when project teams cannot agree on a single scheme to categorize the content.

Navigation

The study of IA informs the navigation design process.

Navigation is about the **user's orientation in the interface, and its main goal is to enable the user to find the information** and functions he or she is looking for and, not only that, to encourage him or her in a direction that might be desirable to him or her.

The basic principles for good navigation are **findability** and **discoverability**.

Navigation

Findability means being able to get to the information you are looking for, or more precisely, how the user is able to localize the information he or she considers relevant.

In a library, for example, it indicates the ease with which one can locate a book one is looking for.

Placing elements consistently within the interface and using standards usually promotes findability.

Navigation

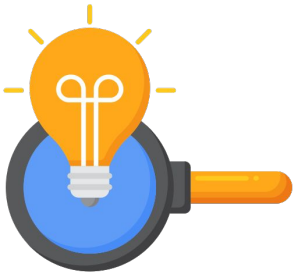
Discoverability refers to the possibility that users have of discovering new information or new features that they were not aware of and were not specifically looking for.

In the library example, it concerns the possibility of discovering a new book that catches one's eye while looking at a shelf or while searching for something else.

The interface often indirectly provides the tools that enable the discovery activity because it can only be evaluated in retrospect. However, it is possible to design for discoverability by placing the new feature, for example, near the most viewed or most relevant items.

Findability vs Discoverability

With Findability the users knows or expects that a certain content or feature will be available.



With Discoverability the users don’t know that the content or functionality is there: they have to discover it!

Users behavior patterns

Users behavior patterns

We have already seen how to identify users needs, motivations and goals (personas, stories…).

But when a user starts an interaction with a system, their actions will be influenced by the information available and the interface in which they are present.

During the last 30 years, Information Architecture experts have recognize that users usually perform a certain set of the same behavior patterns: the flow of the succession of events between user and system.

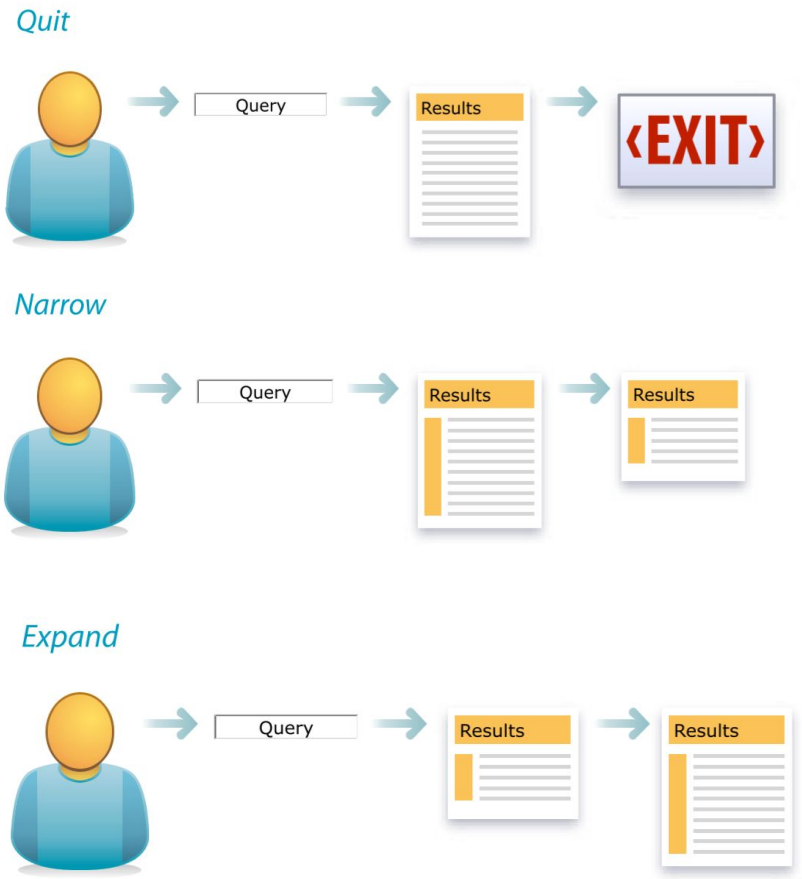
Specifically, in regard to information-seek, different different patterns of behavior have been identified. In the next slides we will see some of them. (Morville Peter, Jeffery Callender. Search Patterns. Design for Discovery. O'Really Media, 2010.)

Information-seeking behavior patterns

1. **Quit**: the user performs a search, sees the results and exits.

2. **Narrow**: the user performs a search, sees the results and refines them using filters, sorting tools or the advanced search.

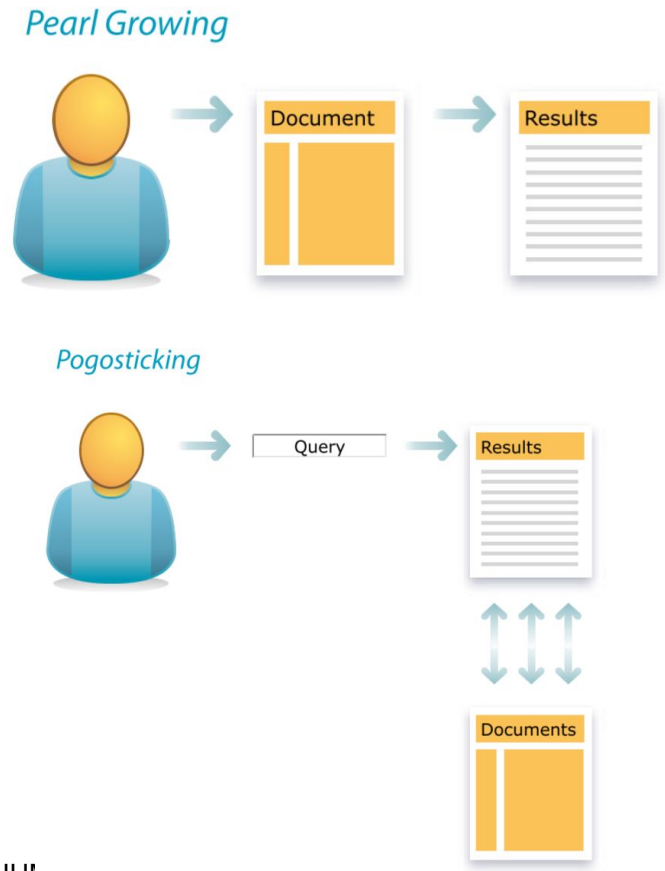
3. **Expand**: the user performs a search, sees the results and expands the scope (displaying related results, etc.).



Information-seeking behavior patterns

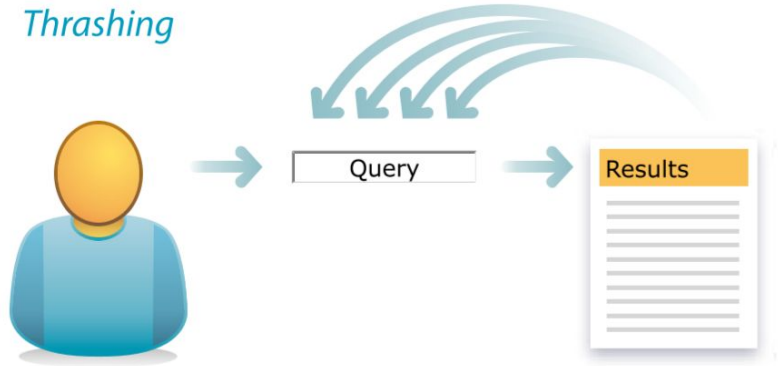
4. **Pearl growing**: the user performs a search, opens one of the results and then opens or uses links within the result (mining topics from the result, typical navigation pattern in Wikipedia).

5. **Pogosticking**: the user performs a search, and then repeats the action of opening a result and returning to the results page.

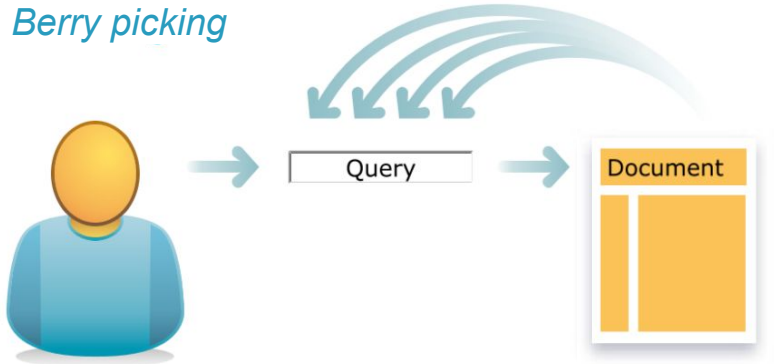


Information-seeking behavior patterns

6. **Thrashing**: the user performs a search, sees the results, and then returns to perform a new search by adding details to the query.



7. **Berry picking**: the user performs a search, opens a result, and from the information received performs a new search refining the query from time to time.



Antipatterns

Some of the patterns we just saw are behaviors that can happens at times:

“That’s to be expected, but when it happens a lot, it’s not sampling; it’s a symptom.” (Morville and Callender)

This is the case of Pogosticking, Thrashing and Berry picking.

If this is a repeated behavior within the interface then it must be considered a symptom of antipattern: **a search pattern produced by a poor Information Architecture and design!**

Design patterns

Design patterns

In parallel and closely related to the behavior patterns, it is possible to identify other possible additional patterns referred, however, directly to the design activity.

This design patterns **have emerged as repeatable solutions to common problems.** They can work like a sort of best practices, or design components that allow you to quickly identify possible features your website or app needs or must provide to the users.

Design patterns

Continuing with the information-seeking activity, a possible set of design patterns could be:

1. **Autocomplete**: completes the search query at the stage when the user formulates it, thus trying to return the searched query as directly as possible.

2. **Autosuggest**: similar to autocomplete but tries to help the user by providing him with even distant (but related) ideas from the query.

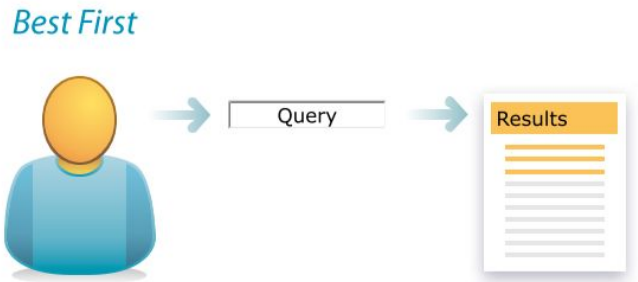
3. **Instant Results**: provides results while the query is being typed.

Information-seeking Design patterns

4. **Did you mean**: after the query is submitted, it provides a hint about the most appropriate result (useful in case of spelling errors for example).

5. **Autocorrect**: instead of suggesting a correction in the search query, it automatically applies it and shows the results.

6. **Best First**: shows as the first results the best ones chosen by an algorithm (by relevance, popularity, date, format, in a customized way for the user specific etc.).

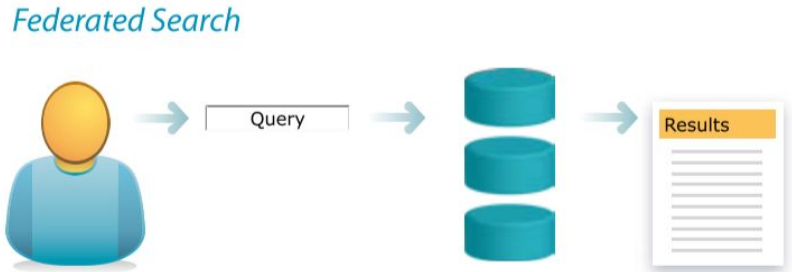


Information-seeking Design patterns

7. **Partial matches**: shows the results that most closely match the query allowing a page with zero results not to be returned.

8. **Related Searches**: shows related searches with the query made that can provide inspiration to the user for further research or help clarify an ambiguous query.

9. Federated Search: allows you to search several databases at once. As a result there will be a variety of results that are difficult to refine.

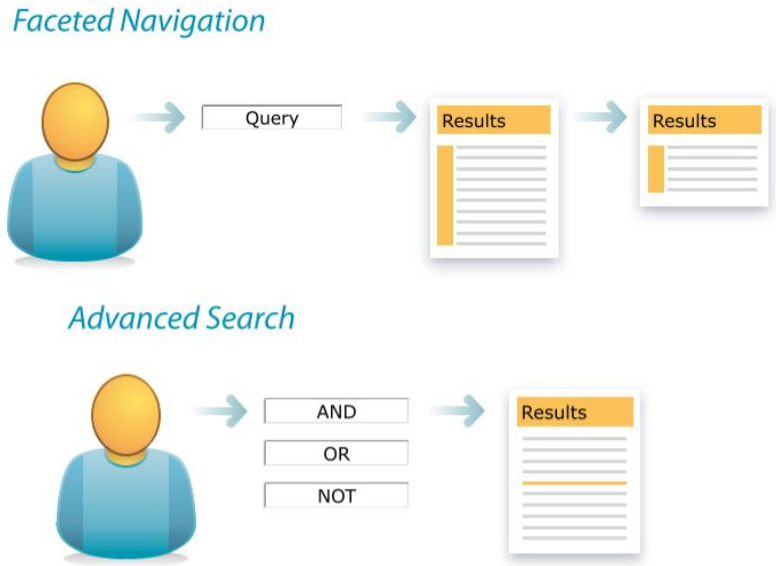


Information-seeking Design patterns

10. **Faceted Navigation**: provides the user with options to refine results through fields regarding different types of metadata (price, color, tags, etc.).

11. **Advanced Search**: allows the user to refine the search before performing it through more or less elaborate options.

12. **Scoped Search**: if content is organized into categories, a dropdown menu can be provided to specify the scope of the search.

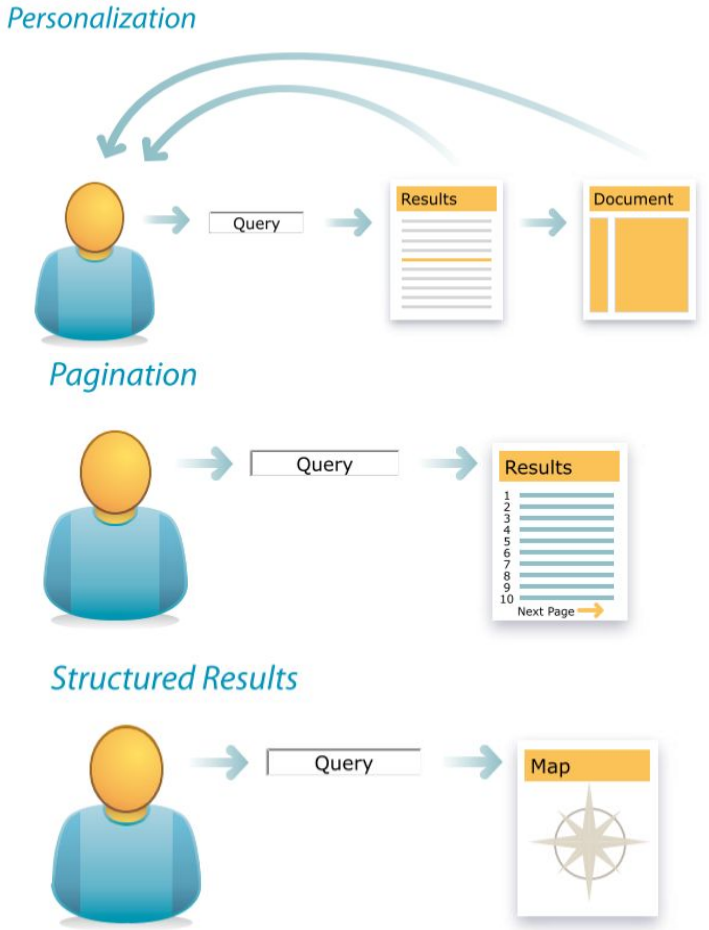


Information-seeking Design patterns

13. **Personalization**: concerns the adaptation of the results shown to the specific user using the system (e.g., recommended results on Amazon or personalized results on Google Maps).

14. **Pagination**: shows a maximum number of results per page. The standard set by Google is ten results per page. Crucial here are the snippets provided for each result (what it is about, available links, etc.).

15. **Structured Results**: each individual result is displayed in the manner most congruent with its content (an address will be represented in a map, a stock market index in a graph, etc.).

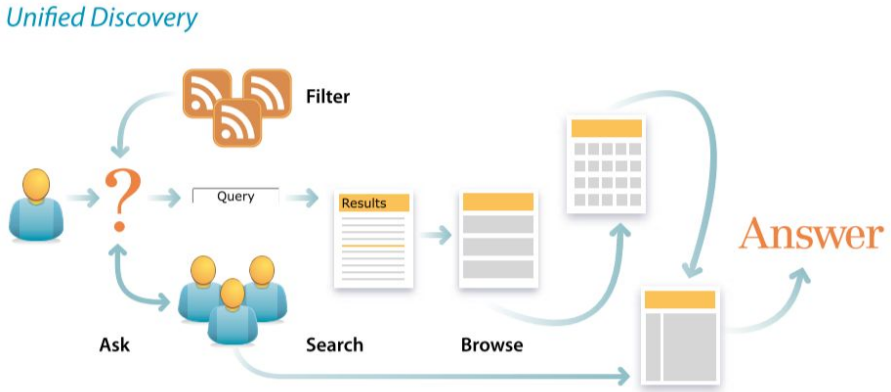


Information-seeking Design patterns

16. **Actionable Results**: depending on its content, each result will provide the ability to interact with it (a phone number that allows a call to be made directly, a video that allows it to be played).

17. **Comparing Results**: allows the user to compare results with each other (for example, the features and price of different products).

18. **Unified Discovery**: the search and refinement modes are combined together. On the same page you can find a search bar, exploration of a browsable taxonomy, faceted navigation, and so on.



Interfaces’ layout and components

The Document Object Model (DOM)

The Document Object Model, usually referred to as the DOM, is an essential part of making websites interactive.

It is an interface that allows a programming language to manipulate the content, structure, and style of a website.

JavaScript is (can be) the client-side scripting language that connects to the DOM in an internet browser

Almost any time a website performs an action, such as rotating between a slideshow of images, displaying an error when a user attempts to submit an incomplete form, or toggling a navigation menu, it is the result of JavaScript (or another web language) accessing and manipulating the DOM.

more info https://www.taniarascia.com/introduction-to-the-dom/

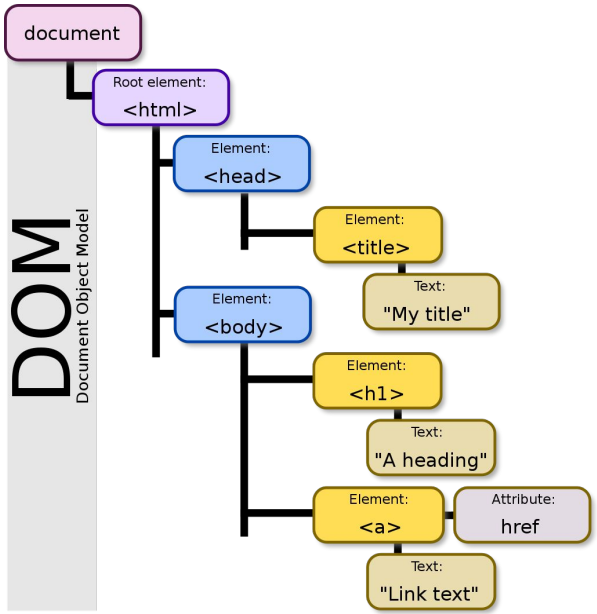
The Document Object Model (DOM)

The DOM is a cross-platform and language-independent interface that treats an XML or HTML document as a tree structure wherein each node is an object representing a part of the document.

The DOM represents a document with a logical tree. Each branch of the tree ends in a node, and each node contains objects.

DOM methods allow programmatic access to the tree; with them one can change the structure, style or content of a document.

Nodes can have event handlers attached to them. Once an event is triggered, the event handlers get executed.



The Document Object Model (DOM)

The principal standardization of the DOM was handled by the World Wide Web Consortium, which last developed a recommendation in 2004.

WHATWG took over the development of the standard, publishing it as a living document. The W3C now publishes stable snapshots of the WHATWG standard.

DOM in Web Browser

To render a document such as a HTML page, most web browsers use an internal model similar to the DOM.

The nodes of every document are organized in a tree structure, called the DOM tree, with the topmost node named as "Document object".

When an HTML page is rendered in browsers, the browser downloads the HTML into local memory and automatically parses it to display the page on screen.

DOM in JavaScript

When a JavaScript web page is loaded, the browser creates a Document Object Model of the page, which is an object oriented representation of an HTML document that acts as an interface between JavaScript and the document itself.

This allows the creation of dynamic web pages, because within a page JavaScript can:

- add, change, and remove any of the HTML elements and attributes
- change any of the CSS styles
- react to all the existing events
- create new events

Interfaces’ components

When designing your interface, try to be consistent and predictable in your choice of interface elements. Whether they are aware of it or not, users have become familiar with elements acting in a certain way, so choosing to adopt those elements when appropriate will help with task completion, efficiency, and satisfaction.

Interface elements include but are not limited to:

- **Input Controls:** checkboxes, radio buttons, dropdown lists, list boxes, buttons, toggles, text fields, date field
- **Navigational Components:** breadcrumb, slider, search field, pagination, slider, tags, icons
- **Informational Components:** tooltips, icons, progress bar, notifications, message boxes, modal windows
- **Containers:** accordion

Continue: here

