Ideation Tool for Creating Collectively Intelligent Systems

by

Abigail Michelle Russell

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Masters of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

Author .....
Department of Electrical Engineering and Computer Science
May 24, 2019

Certified by .....
Thomas Malone
Professor
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Ideation Tool for Creating Collectively Intelligent Systems

by

Abigail Michelle Russell

Submitted to the Department of Electrical Engineering and Computer Science
on May 24, 2019, in partial fulfillment of the
requirements for the degree of
Masters of Engineering in Computer Science and Engineering

Abstract

In this project, I build tools for helping imagine and create collectively intelligent systems. The main ideation tool allows the user to map out and navigate complicated systems in a structured way, while providing prompts to nudge the user into looking at the system in a new way. I also centralize a database of case studies about collectively intelligent systems.

Thesis Supervisor: Thomas Malone
Title: Professor

Acknowledgments

Thanks to many members of the Center for Collective Intelligence for meeting with me and giving feedback, including Kathleen Kennedy, Annalyn Bachmann, and Leila Snyder.

Thanks especially to Thomas Malone, who supervised this thesis. Much of the work here builds off of previous work he has done. With an incredibly busy schedule, he still found time for brainstorming sessions and mentoring for my career beyond the thesis.

Contents

List of Figures

Chapter 1

Introduction

Collectively intelligent systems are groups of humans and/or machines that work together in ways that seem intelligent [1](p. 19). They can often lead to more efficient, creative, or robust solutions. Hyper-connected humans and the rise of artificial intelligence are making collective intelligence systems more valuable and more possible. Leaders of organizations are increasingly looking for insight into how to structure organizations to take advantage of this trend.

The MIT Center for Collective Intelligence aims to both study collectively intelligent systems and to apply these systems in new ways. This thesis work supports these goals. The project aims to first bring together knowledge of collective intelligence by building a library of case studies, and then to help users visually map out their organizations and processes, so that they can zoom in on where best to apply collective intelligence.

1.1 Superminds

A core thesis of the Center for Collective Intelligence is that business success will increasingly come from more intelligent organizations, not just more productive ones. A supermind is another name for a collectively intelligent system [1]. Superminds provide a way of thinking about how organizations can become more intelligent.

Figure 1-1: Supermind Icons [8]

By hyperconnecting large groups of humans and/or computers together in organized ways, systems can take advantage of the different strengths of the many participants. Superminds can be categorized into several buckets, illustrated in Figure 1-1, describing how they make decisions.

1. **Hierarchies:** Where group decisions are made by delegating them to individuals in the group.

2. **Democracies:** Where group decisions are made by voting.

3. **Markets:** Where group decisions are the sum of all pairwise agreements between buyers and sellers.

4. **Communities:** Where group decisions are made by a kind of informal consensus based on shared norms and reputations.

5. **Ecosystems:** Where group decisions are made by the law of the jungle and survival of the fittest.

The tasks that superminds perform can be broken down into five basic cognitive processes. In order to take an action, you must first Decide what to do. To make a decision, various options for action must be Created. The process of deciding and creating become better when you can Sense the world and Remember the past. The ability to Learn improves the entire process, allowing you to take better action [1].

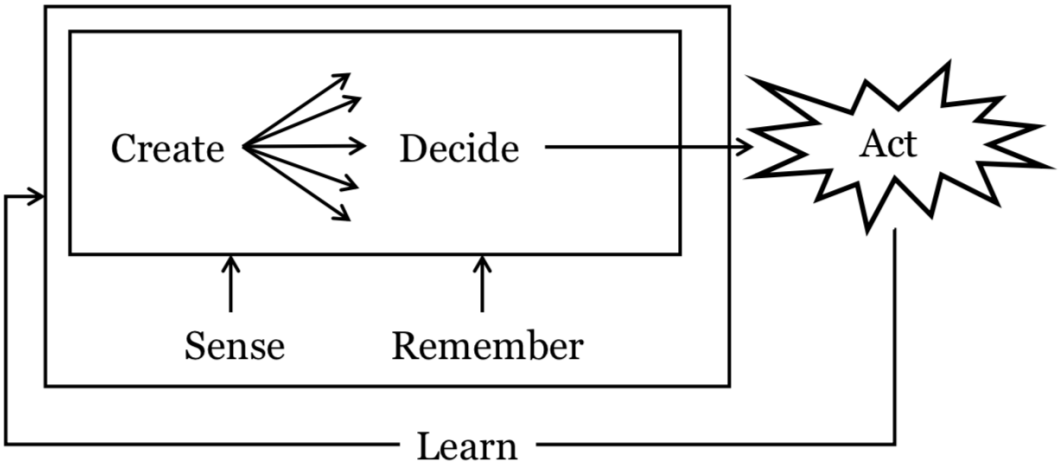


Figure 1-2: Cognitive Processes [1]

1.2 This Project

To motivate the problem I am trying to solve, let us look at a project that the center has recently been involved in: Radically redesign the Japanese Healthcare System. How would you go about doing that? Systems like this are large and complex. There are many different types of superminds and ways they can be applied. It is challenging to focus on one area and crystallize the changes to the system. What's more, one can easily get stuck in one way of looking at the system, missing opportunity.

The first step for solving any problem is to break it down into its parts. We could look at how tasks can be subdivided when delivering care. Here are just two ways we might break that down:

1. **By Treatment Phase**
    - Prevent ⇒
    - Diagnose ⇒
    - Treat
2. **By Actor**

- Patient
- Physician
- Hospital

There are hundreds of ways to break down this system. There are even more subsystems to zoom in on. One might consider the process of developing, manufacturing, advertising, prescribing, and delivering drugs. There is the regulatory structure that dictates everything from hospital hallway widths to the language used to describe prescription drugs. These types of problems are hard, but they're also perfectly suited for the benefits of superminds.

In order to facilitate the conceptualization process, I put forth an ideation tool. This tool functionally allows the user to structure their system, thoughts, etc. I build the interface with built-in prompts to help the user think about their system in different ways. I attempt to use the Center's knowledge of already-existing superminds to suggest to the user how they can be applied in their system. To support this work I create a database of case studies of superminds in practice.

1.3 Previous Work

Much of the work in this project builds on, is inspired by, or supports the work done at the Center for Collective Intelligence.

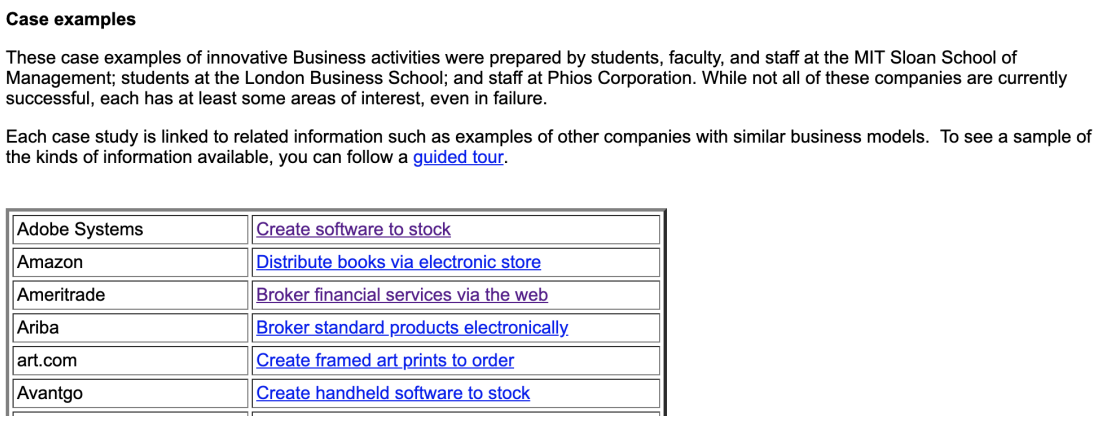
1.3.1 Superminds

This project builds on previous investigations and studies from the Center for Collective Intelligence. Notable work includes *Superminds* [1], which collects examples of superminds and breaks them down into categories. A supermind is another name for a collectively intelligent system. The various categories and subcategories of superminds, outlined in the previous section, are valuable tools when navigating our database.

1.3.2 Business Processs Handbook:

The Business Process Handbook [2] previously imagined new ways to break down and explore business processes. This guidebook introduces several concepts that will be useful to incorporate into the interface.

Part of that project was to build a library of case studies. That library is viewable at http://process.mit.edu/Info/CaseLinks.asp. We build on some of the concepts designed into that database, as well as learn from their usability problems and bringing in design tools that were not available when it was built.



Case examples

These case examples of innovative Business activities were prepared by students, faculty, and staff at the MIT Sloan School of Management; students at the London Business School; and staff at Phios Corporation. While not all of these companies are currently successful, each has at least some areas of interest, even in failure.

Each case study is linked to related information such as examples of other companies with similar business models. To see a sample of the kinds of information available, you can follow a guided tour.

Adobe Systems	Create software to stock
Amazon	Distribute books via electronic store
Ameritrade	Broker financial services via the web
Ariba	Broker standard products electronically
art.com	Create framed art prints to order
Avantgo	Create handheld software to stock

Figure 1-3: Screen Shot from the Business Process Handbook Case Library [2]

1.3.3 Smalltalk Browser

The SmallTalk browser is an IDE (Integrated Development Environment) optimized for reading code. It forces the user to define a code structure in a semantically meaningful way, similar to what I want to achieve with the interface [7].

The browser has a lot of value structurally, but lacks many of the UI improvements of the modern age.

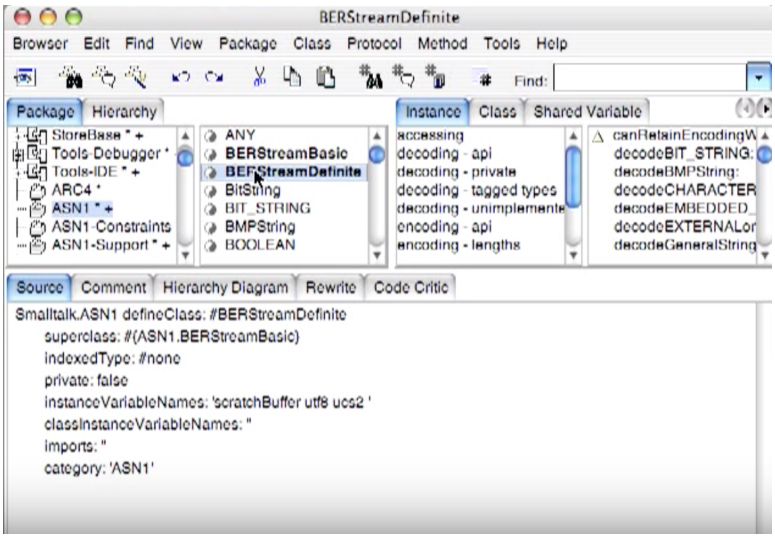


Figure 1-4: A Screenshot of the SmallTalk Browser

1.4 Design Goals

- **Usability:** The most important design goal of the system is usability. User-focused design means designing the system to be as intuitive, convenient, and useful to the user as possible. This system is only useful and interesting in relation to how a person uses it. Particularly, the ideation tool needs to be as convenient as possible, so that the user is focused on the problems they are trying to solve and not on using the tool.

- **Flexibility:** Flexibility is the second most important design goal. We expect users to use the system in ways we have not anticipated. Especially in the case of the ideation tool, we want the interface to be flexible enough to be used to structure many different systems and problems. Flexibility in implementation is important too. The system should be easy to change and be built on for future needs.

- **Simplicity:** Simplicity is an important goal, as it supports the two most important design goals. Simplicity of implementation will support flexibility. A

clean, simple interface design will make the system more intuitive and usable.

- **Performance:** System performance is a minor design goal. The system needs to be performant enough that it can smoothly pull data from the database, process it, and update the interface without causing any noticeable wait time for the user. Beyond the threshold of inconvenience, however, performance is not an important part of the system.

- **Scalability:** Scalability is generally not a design goal of this system.

1.5 Technical Infrastructure

I chose to build the system on top of technologies that support the needs of the system as well as fit into the design goals laid out in the previous section. This section goes into detail about technical design decisions.

1.5.1 ReactJS

ReactJS is a open source JavaScript library for creating user interface software. I build out the front-end interface using React. React allows the creation of reusable UI components. This allows the system to be built in line with software best practices, keeping the code *safe from bugs*, *ready for change*, and *easy to understand*.

Many of my interface designs involved one single page with many changing components. React allows the system to dynamically update and change components without the user needing to reload the page. This, along with the visual libraries supporting the platform, make the visual components of our system relatively easy to implement.

The use of React highly supports the design goals. React is very simple in implementation, making the code easy for other people to understand and change. Long-term flexibility of the system relies on easily modified code. React is exceptionally performant. Though scalability is not explicitly a design goal of the system, React

has the added bonus of scalability. Additional information about ReactJS can be found in their documentation [4].

1.5.2 Firebase

Firebase is a mobile and web applications development platform owned by Google. The Firebase platform offers backend as a service. Firebase offers many components needed for the application, such as database services and web hosting. Using Firebase will allow the application to be built without the need for someone to have ownership over and maintain the system's own servers.

The use of Firebase aligns well with the goals of this project, especially the simplicity of implementation. Firebase easily integrates with react applications. There is minimum code and configuration needed, but there is enough customization to allow for a lot of flexibility. The use of Firebase does limit our scale at:

- 100 Simultaneous Connections
- 1 GB Stored
- 10 GB Downloaded /Month

In the event that the project grows beyond this capability, the Center will have to change the provider or pay for an upgrade of service.

Additional benefits to using this platform include an authentication service, keeping unauthorized actors from writing to and reading from our database, and the fault tolerance of Google's cloud infrastructure.

Realtime Database

Firebase's realtime database stores data on the cloud and synchronizes that data across applications and clients. Data is stored in a NoSQL database, in a flat structure similar to a large JSON (JavaScript Object Notation) object. Firebase uses local cacheing to serve and store data in case of a service outage or lost network connection.

The data is synchronized when the user comes back online. This lends our system fault-tolerance.

Hosting

I similarly chose Firebase as the platform to host the application because of its simplicity. Firebase hosting is simple to deploy and also works well with the Firebase Database platform.

Additional information about the Firebase platform can be found in their documentation [5].

Chapter 2

System Overview

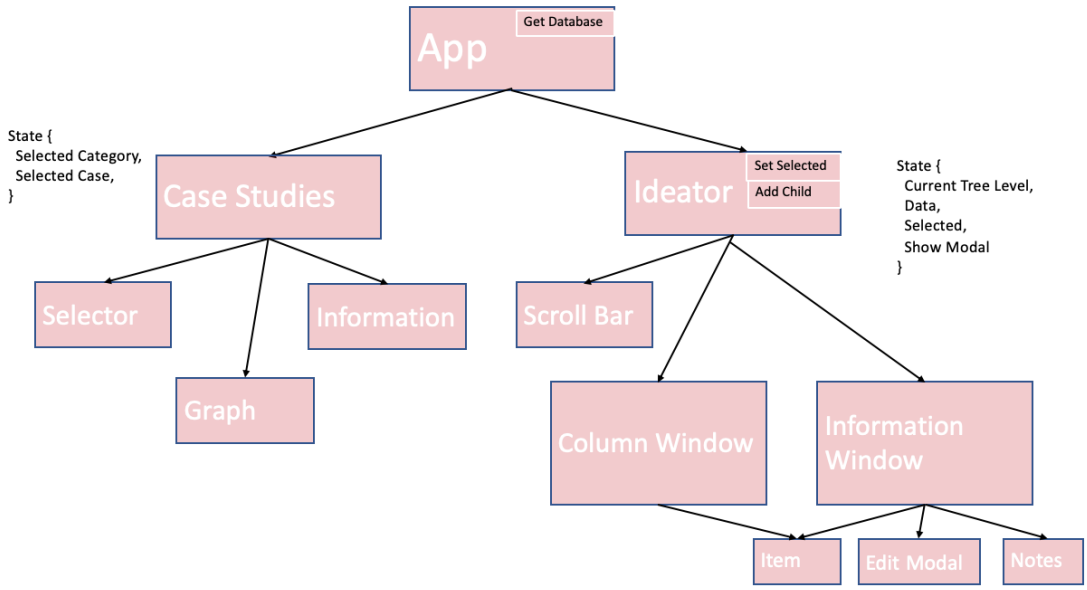


Figure 2-1: System Diagram

This illustration shows the hierarchy of major components of our system. Parent functions are usable by children, and we show some crucial ones such as Get Database in the App component.

A child can use its parent's state information, and when the parent's state updates,

the child component does as well. This is the reactivity of ReactJS. The system diagram shows some state information, important for reactive updates of children components.

Chapter 3

Ideation Tool

In this section I break down the ideation tool, describe its development process, and evaluate it based on our design goals and user interface design principles.

3.1 Design History

When designing the ideation tool, I iterated through several design concepts. Below I illustrate some of those design concepts.

3.1.1 Outline Format

This design was by far the simplest to understand and the most flexible. This interface prompts the user with structured questions, structured in a hierarchical tree. A concept interface is shown in Figure 3-1.

Figure 3-1: Outline Concept

3.1.2 Stakeholder Map

This concept was inspired by *stakeholder mapping*, an important part of stakeholder analysis. Stakeholder analysis is frequently used by managers to assess the impact of a project on different stakeholders. There are many different mapping techniques to visualize and aid in stakeholder analysis. We draw inspiration from the *community of stakeholders* mapping concept illustrated in figure 3-2. The key part of this concept is mapping how stakeholders interact with one another.

Figure 3-2: Community of stakeholders [6]

One way we could look at a system one might map using my tool is through the stakeholders: who are the actors in the system? By mapping out the existing interactions between different stakeholders, one can see which actors or groups are lacking in communication. The concept design shown in Figure 3-3 embodies that idea. The proposed interface would let users define the stakeholders in the system and map out their interactions. The ultimate goal of this design was to find stakeholders that aren't connected together, and then to show the user case studies where those groups had been connected in a supermind. For example, in Figure 3-3 the customers are not connected to each other, nor are they connecting to the employees. The interface might suggest the Glossier Slack Case. In this case, Glossier created a slack channel for their top customers to talk to each other. The interactions between the customers in that channel lead to the creation of new products.

This was an interesting line of thinking, but the purpose of this design did not materialize clearly. This idea only applied to a narrow set of situations.

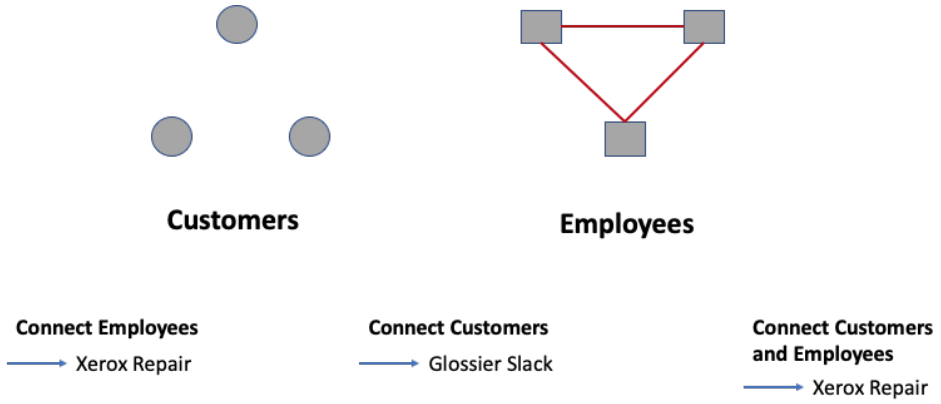


Figure 3-3: StakeHolder Map Concept

3.1.3 Cognitive Process Map

This design also centers on the actors in the system. Instead of mapping the stakeholder interactions, this interface design mapped the actions they took. The main idea here was to match up actions with supermind examples at the same intersection in the cognitive process cycle described in Figure 1-3.

The concept is shown in Figure 3-4: by looking at functions of a group, the interface might pinpoint useful supermind types to apply. It could also be useful to see where some groups could be taking on more action.

3.1.4 Goal Oriented

The goal-oriented interface organized cases first by the ultimate improvement the supermind caused on the system. The example in Figure 3-5 shows how case studies can be ordered by the goal of the supermind. In the Wikipedia case, the organization lowered the cost of content moderation by enlisting community members and bots to police content. This case might be more applicable to someone whose primary objective is to lower cost and increase scale.

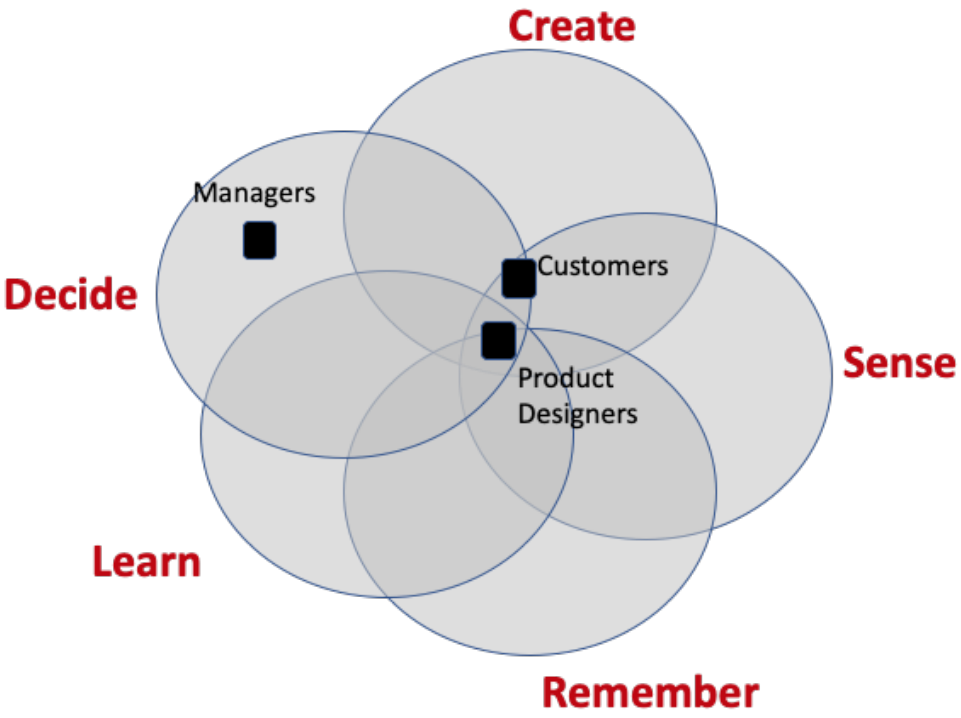


Figure 3-4: Cognitive Process Map Concept

Goal:	Case:	Supermind:
↓ COSTS	Wikipedia	Community
↑ EFFECTIVENESS	Pirate Parties	Democracy
↑ EFFECTIVENESS	Google	Democracy

Figure 3-5: Goal Oriented Concept

Though all of these design concepts lead to interesting exploration sapces, many of them are very applicable to some problems or systems but lack the flexibility to work with every system. I decided to focus on the outline format, as it best fit the design goals of the system. For this concept, a flexible tree-like data structure, a

simple visual design, and an easy to use interface needed to be created.

3.2 Design

The first iteration of my design had three separate views, one dedicated to the outline format, one to the cognitive process of the action selected, and one to suggested superminds and case studies. Figure 3-6 shows a screenshot of the interface prototype. The design came about as a visualization of the outline in Figure 3-1.

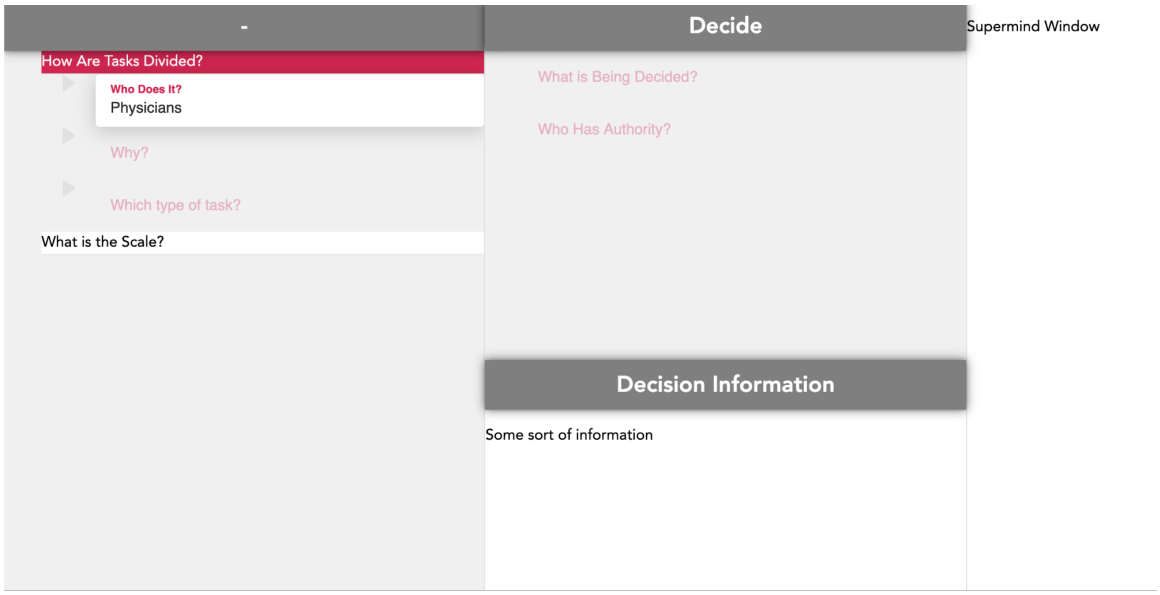


Figure 3-6: Ideation Tool Version 1

This design had unnecessary complexity baked in. Furthermore, it could be designed in a much more flexible way without sacrificing function. The second iteration was designed with the outline format completely in mind. Still with three separate views, but this time each view is a layer in our tree-data structure. The tree can be of arbitrary depth. Though it was designed to accommodate the question-answer format of the outline, the tree data can be arbitrarily any text. Because of the likelihood of deep-detailed trees, I added a scroll bar at the top for usability purposes. The scroll bar allows the user to intuitively drag the bar to move up and down the tree-changing the three views. It dynamically changes size depending on the depth of the

tree, and can be used to keep track of the users position in the tree. Figure 3-7 shows a screen shot of this version. Figure 3-8 shows the same version laid out with some of the main components shown in our system diagram in Figure 2-1.

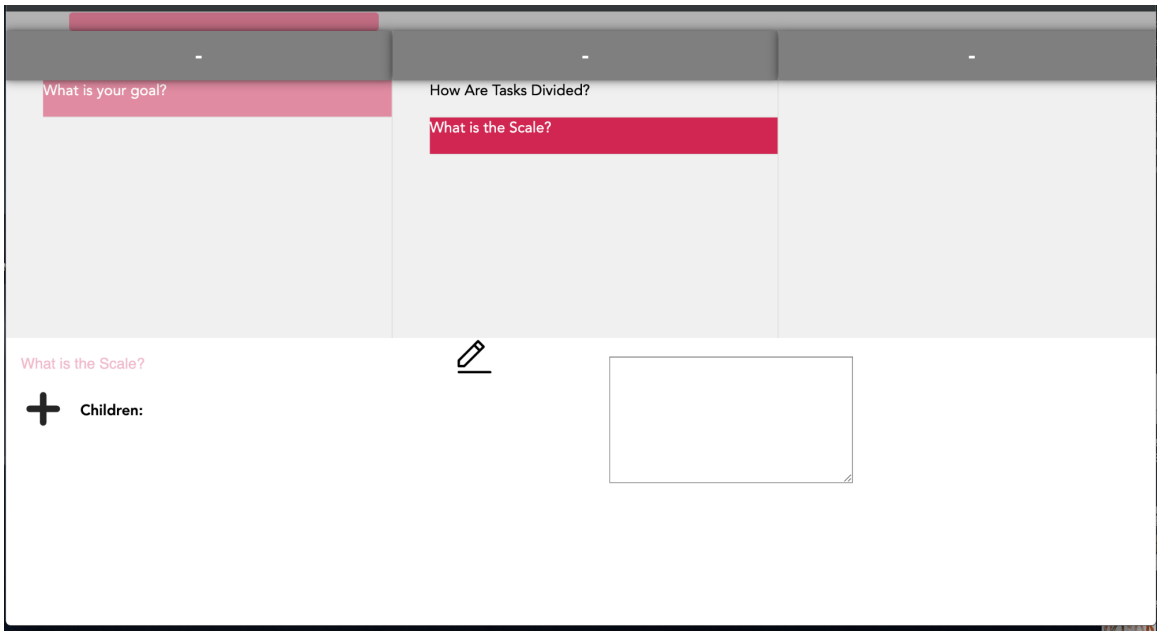


Figure 3-7: Ideation Tool Version 2

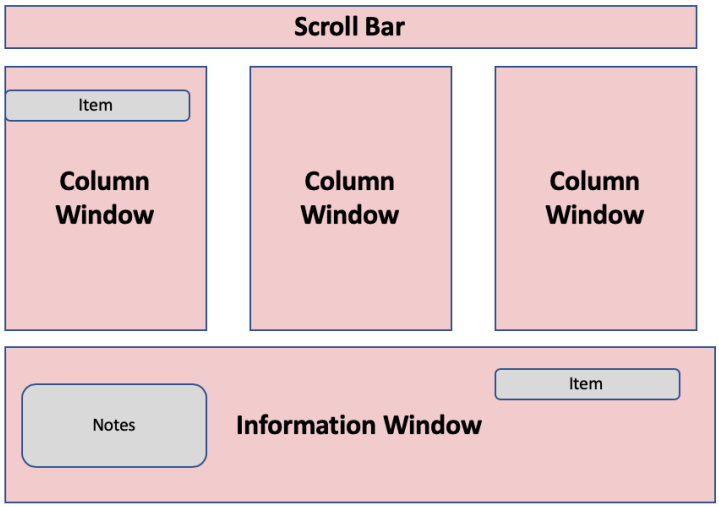


Figure 3-8: Component Layout

For further iterations, I focused on usability improvements. A frame of the improved interface is show in Figure 3-9. The main points of concern were how intuitive and easy to learn the interface is. I added components, like arrows in the scroll bar, to make the interface more easily understood. In this interface, the main selected item is always focused in the middle, with its children always on the right and parents on the left. The children in the bottom information window are located under the children in the upper window, for consistency.

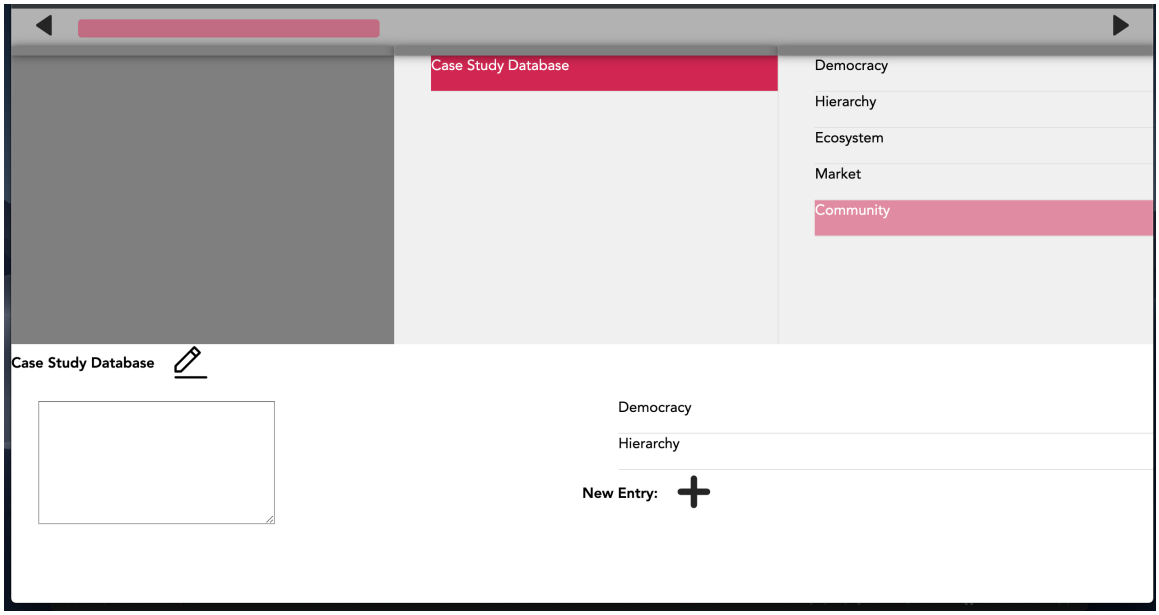


Figure 3-9: Ideation Tool Version 3

3.3 Efficiency Enhancement

To enhance the usability of the interface, I seek to minimize the time and number of clicks it takes a user to complete a task. For the prototype, the point-and-click method of selecting components and navigating the interface was primarily used. For the final version, I seek to maximize the functionality of the interface using the keyboard instead of the mouse.

I make navigation more efficient by introducing arrow-key short cuts to move focus between different items. Right and left arrows move the user to a different column,

and the up and down arrows move the user to a different item in the same column. I also make adding additional items more efficient by allowing the user to create a new child item with the enter key.

To test these improvements, I ran a user test. I asked participants to use the ideation tool to recreate the tree in Figure 4-2 using the interface. I ran the test using three participants and averaged the time it took them to complete the task.

Original Interface	Improved Interface	Speedup
114 sec	83 sec	27.2%

3.4 Data Structures

The data for the ideation tool is stored in a Firebase database. For each data set we store the name of the dataset and its tree depth. Since firebase delivers data in a flat object, I architect the data structures for that environment. Storing more dense data points is not a problem, as likely there is much more raw storage space than needed. It is important to avoid having to iterate over data sets when not necessary. Since the interface is designed to dynamically update with user interaction, quick processing of data is very important. A slow algorithm for data updates could cause usability and performance issues as the data grows larger. I want to be able to traverse the data tree with little overhead, so I use a doubly-linked list structure to hold the data. The hash table keyed by IDs ensures an O(children) processing time when traversing the tree downwards and an O(1) processing time when traversing the tree upwards (since each data point can only have one parent). In general, if information can be stored over calculating it I do, as in the case of the depth that data point is in the tree.

```
meta: {
  "deepest_level": 4,
  "name": "Example Dataset"
},
0: {
  "tagline": "What is your goal?",
  "id": 0,
  "level": 0,
  "parent": "",
  "children": [1],
  "notes": "",
},
1: {
  "tagline": "How Are Tasks Divided?",
  "id": 1,
  "level": 1,
  "parent": 0,
  "children": [2, 3],
  "notes": "",
},
...
```

Figure 3-10: Ideation Tool Database Data Structure

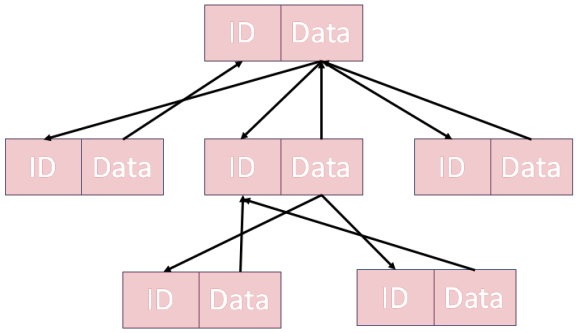


Figure 3-11: Doubly Linked List

3.5 Analysis

I attempt to evaluate our interface in regards to user interface design principles and our design goals.

3.5.1 User Interface Design Principles

Learnability

The interface can functionally be understood intuitively or easily picked up on through the exploration process. More user testing is needed to determine which parts of the interface are confusing to the user.

Efficiency

Through the time minimization techniques discussed in Section 3.4, efficiency has been prioritized and maximized.

Consistency

Consistency was prioritized throughout the interface, code, and paper. For the interface, there are no major functional or aesthetic inconsistencies that could lead the user to err. On the implementation side, consistent naming schemes, color schemes, and code structure were used.

Safety

The interface does not make it easy to make a major mistake, but also does not make it easy to recover state in the case of a error on the user side. The safety of the interface could be improved upon.

Function

The interface functions as intended, though its scope could be expanded.

3.5.2 Design Goals

Usability

As shown in Sections 3.6.1 (UI Design Principles) and 3.4 (Efficiency Enhancements), this interface is designed with usability as the main driving factor.

Simplicity

Simplicity, both in implementation and in user experience, are generally achieved.

Flexibility

Flexibility is prioritized, even over functionality, as shown in section 3.1.

Performance

Performance is achieved in terms of the boundaries we defined in Section 3.6.2. Further testing is needed to determine the scalability limits of this performance.

Chapter 4

Case Study Database

4.1 Introduction

Many members of the Center for Collective Intelligence expressed the value in simply compiling case studies together in a centralized place. As mentioned in Section 1.3.1, previous efforts by the lab have sourced many great examples of collectively intelligent systems. These resources are not centralized and inconsistently formatted, however. The biggest value add of this part of the project is to simply compile and organize new and old case studies of collectively intelligent systems. A secondary goal of this interface is to inspire the user to think about these examples in new ways and to find connections between cases that might seem very different. I thus design a visual interface for exploration through the case study database.

4.1.1 Initial Designs

4.1.2 Wikidata

I initially planned to build the system on top of the wikidata framework. Wikidata is an open knowledge base that can be read and edited by both humans and machines. It centrally stores structured data, in a relational way. Ultimately, using the wikidata framework came with too many drawbacks. Since it is an open platform, any actor can corrupt our data. This vulnerability takes away from the fault-tolerance of the

system. There is also no way to restrict viewing access. Using wikidata was not as simple in implementation or as flexible as Firebase.

4.1.3 Graph Explorer

To visually explore the relationship between case studies, I set out to represent them graphically. Figure 4-1 shows an early iteration of this design.

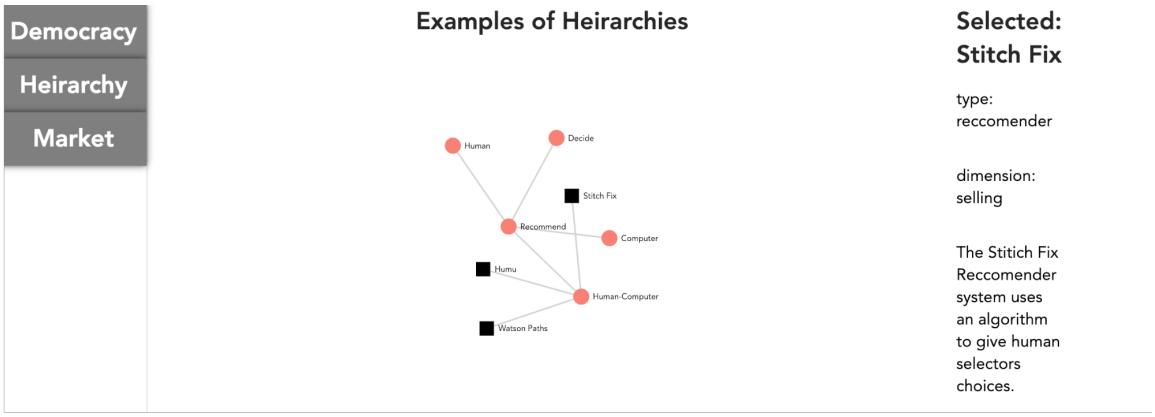


Figure 4-1: Case Study Interface Initial Design

This concept proved to be too unstructured. The next iteration illustrated in Figure 4-2 adds structure to the graph, based on categorization from *Superminds* [1].

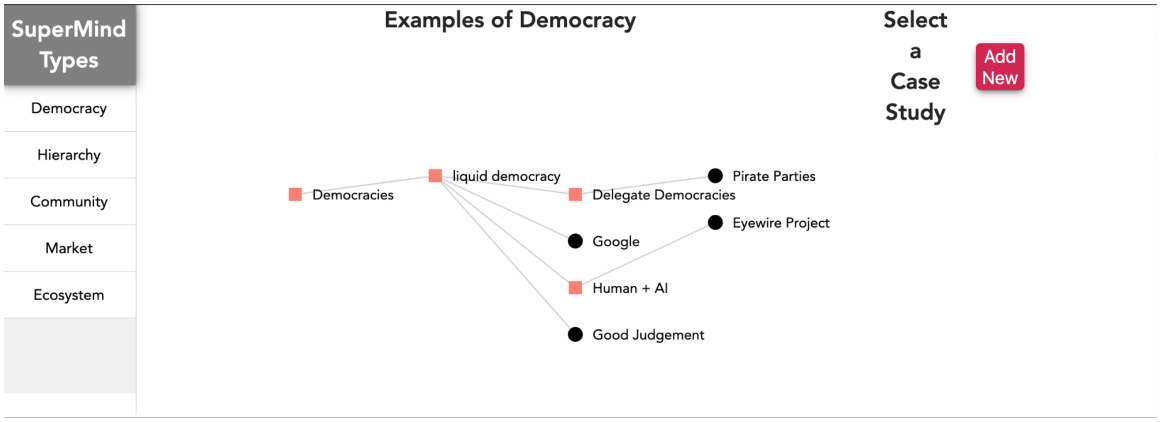


Figure 4-2: Case Study Interface Initial Design 2

4.2 Analysis

I attempt to evaluate our interface in regards to user interface design principles and our design goals.

4.2.1 User Interface Design Principles

Learnability

This interface does not have many actions the user can take, and most of them are clearly labeled buttons or clickable components, which is easily learned through exploration.

Efficiency

Efficiency was not prioritized in this design.

Consistency

For the interface, there are no major functional or aesthetic inconsistencies that could lead the user to err. On the implementation side, consistent naming schemes, color schemes, and code structure were used.

Safety

The interface does not make it easy to make a major mistake, but also does not make it easy to recover state in the case of a error on the user side. The safety of the interface could be improved upon.

Function

The interface functions as intended.

Though an interesting concept, exploring case studies graphically turned out not to be as useful as thought. The ideation tool is flexible enough to hold the case

study database in a similarly structured way. Navigation through the ideation tool is sufficient.

Chapter 5

Conclusion

From experimenting with representing the case study database graphically and iterating through the ideation tool, I learned that making interfaces more flexible and simple is often better. There is a lot more work that can be done on this project, but I have laid a solid foundation that can be built on by others in exploring a lot of interesting ideas put forth by the Center for Collective Intelligence.

5.1 Looking Forward

There are many usability improvements and added functionality that I plan to add to this interface to make it more useful to the Center. More user testing on features, usability, and efficiency enhancements will give more direction and insight. I also plan on exploring more use cases beyond the case study database and organization mapping.

Bibliography

[1] Malone, Thomas: Superminds. 2018

[2] Malone, Thomas, Crowstone, Kevin, Herman, George: Organizing Business Knowledge: The MIT Process Handbook. 2003.

[3] Wikidata. https://www.wikidata.org/wiki/Wikidata:Main_Page.

[4] ReactJS Documentation. https://reactjs.org/

[5] Firebase Documentation. https://firebase.google.com/products/realtime-database

[6] https://evolveea.com/cities-coming-to-life/

[7] http://onsmalltalk.com/on-the-smalltalk-browser

[8] Malone, Thomas W. Superminds: The surprising power of people and computers thinking together. MIT Book Launch Event, Cambridge, MA, May 21, 2018 [figure is from slide used in presentation].