White Paper

Human Centered Design

Creating software for diverse end users

By Liviu Catuneanu Cica and Neil Kenney

MACEDON TECHNOLOGIES

Table of Contents

What is HCD & Why Do We Care? 3

The Phases of HCD 3

HCD and Disabilities 5

Agile and HCD 5

How Macedon Uses HCD 6

Conclusion 8

What is HCD and Why do we Care?

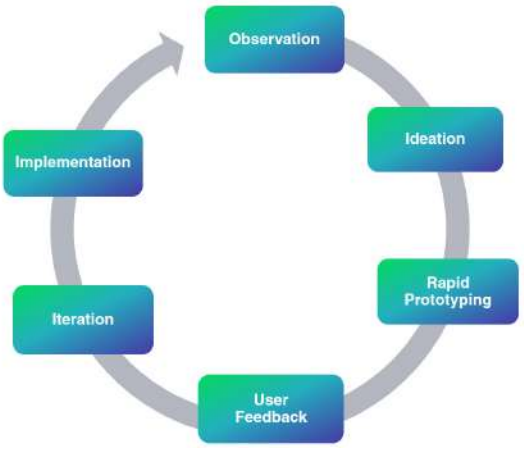
Human Centered Design (HCD) is a design philosophy that involves factoring the experiences, perspectives, and needs of the people who will be using a tool into the development of a product. In the case of software, this means centering software design around intuitive and easy to understand interfaces that encompass a wide range of ability levels. Software designed without user experience as a priority is often abandoned for more appealing alternatives. Employees forced to use poorly-designed software operate at decreased efficiency. Well-designed software should increase efficiency and speed up task completion, providing a return on investment. Using HCD to gain an understanding of how people perform their jobs is paramount to creating well-designed software.

HCD focuses on the worker's needs and goals to determine the features and processes that will be the most beneficial. By incorporating constant and consistent feedback, HCD considers how the end product should holistically work and feel, instead of getting caught up in smaller details. Taking a comprehensive approach to design facilitates "big-picture" thinking, promotes interorganizational collaboration, and often reveals creative approaches to design challenges.

Consider the example of a piece of software created to assist with documenting financial transactions. The software's requirements stated that users must be able to enter details about a relevant account so that transactions can be properly linked to that account. On its surface, this seems like a fairly innocuous technical requirement. Unfortunately, the software was designed so that the same information must be re-entered for every transaction. While the software satisfies the requirement as written, its poor design creates redundant work. Through the use of HCD, a prototype leading to swift feedback would have revealed the redundancy and resulted in a better, more efficient product.

The Phases of HCD

There are 6 main phases of the HCD lifecycle: Observation, Ideation, Rapid Prototyping, User Feedback, Iteration, and Implementation. These phases are based on the six phases of IDEO's design process, as outline by UserTesting.[1]



Before development can begin, designers must first know the problems the software needs to solve; this is where **Observation** comes in. One way to perform this is to schedule a meeting with people who currently work within the process in question and have them walk through

how they currently complete their tasks. The designer should note the most useful features in the existing software, as well as pain points that should be addressed by the new product. If in-person work sessions are not feasible, video recordings or written walkthroughs of the routines can also be effective. Learning directly from the people for whom the software will be designed gives designers the ability to fully understand their needs.

Only when the workers' needs are fully understood will you have the ability to start the **Ideation** phase. This is when personas - conceptual definitions of the stakeholders that are expected to interact with the software - and journey maps - the steps that someone must take in order to complete a job - are made, and we can then start to map out what the software could look like. This will lead to software with a solid foundation that is backed by an understanding of who the users are and what steps they need to take to perform their job.

Using all of the information gathered during the first two phases, user stories can be formed to facilitate **Rapid Prototyping.** During the Rapid Prototyping phase, designers will quickly make workable prototypes that closely approximate the final software and can be tested by stakeholders as quickly as possible. When prototyping, there is more to consider than just the user interface; associated processes should also assist your workers in completing their work. Requirements for complex business processes can be complicated to accurately document, leading to a potentially flawed process design. Through HCD practices, the stakeholders involved should experience prototyped workflows to ensure that the software is providing the intended value.

Once the prototypes are created, we enter the next phase of HCD, **User Feedback**. It is extremely important to give participants time to experience the prototypes and provide feedback on their experiences. Even rough prototypes can spark rich conversations that positively influence software's effectiveness. Getting feedback can be accomplished in many ways. For example, the designers can schedule a live session with stakeholders to get direct verbal feedback and spark conversations. If that's not possible, the prototypes can also be sent offline, accompanied by a questionnaire to capture their feedback.

After feedback is collected, the software enters the next phase, **Iteration**. During this phase, the designers reflect on the feedback received, and continue to participate in coding/testing cycles, fine-tuning the software. The team needs to take into account how much time is available to create the software as well as which feedback items will provide the most value to the stakeholders who will be using the system. The goal is to address as much of the feedback as possible while taking value, time, and resource constraints into consideration.

During the final stage, **Implementation,** the team releases the "final" version of the software. The development cycle can then begin anew, creating updates

and applying new features to the application.

HCD and Disabilities

While HCD can have a dramatic effect on work productivity, it can even more dramatically affect work performed by individuals with disabilities. In some cases, such as federal projects, HCD can be used to make sure a new system has met 504 compliance requirements. In others, it can be an ethical or social consideration to help differently abled individuals interact with a new system.

In general, there are four main categories of disabilities that HCD designers take into consideration when building systems: intellectual disabilities, physical disabilities, mental illness, and sensory disabilities. These disabilities can be thought of as restricting a user's ability to manipulate the software, communicating information to the software, and understanding information conveyed by the software. Those who employ an HCD approach use specific design patterns to address all of these disabilities. The following graphic shows some tips to design software for different types of disabilities.

Intellectual Disabilities
Use fonts that are designed for users with dyslexia
Add icons for users who are unable to read labels
Reduce user's reliance on memory by displaying relevant information the user previously selected
Use dropdowns with optional free text sections

Physical Disabilities
Avoid designs with click and drag functionality
Avoid keyboard shortcuts involving multiple keys
Make buttons large and far-spaced to avoid misclicks
Ensure the system behaves identically between single and double clicks

Mental Illnesses
Simplify interfaces
Display realistic images of people (as opposed to those that show exaggerated emotions)
Showcase diversity in images to foster a sense of community and reduce feelings of loneliness and isolation

Sensory Disabilities
Provide colorblind modes or avoid color combinations for common types of colorblindness
Provide alt text for all images
Provide subtitles of any audible Language

A large portion of the tips outlined are not overly complicated to implement but provide great help to those with disabilities that want or need to interact with software.

Agile and HCD

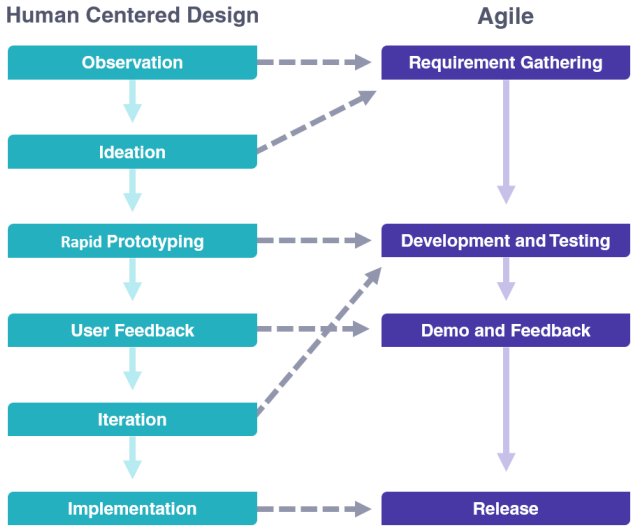
Knowing the phases of HCD alone is not sufficient to understand how to incorporate human centered design into software development. Tying them to a software development methodology helps to bridge the gap between theory and application. The HCD phases align well with the Agile software development methodology.

HCD Observation and Ideation go hand-in-hand with the requirement gathering phase of Agile. Understanding who the stakeholders are and working with them to create requirements ensures value upon completion.

HCD Rapid Prototyping and User Feedback can also be included as part of the requirements gathering process when thinking of the prototypes as mockups. If mockups are created and presented to users for feedback before development on the stories begins, those stories have a much stronger definition for what needs to be accomplished. This can lead to a more complete Definition of Ready for project teams who embrace HCD practices.

Prototyping can be considered part of development because in Agile the goal is to work in relatively short periods of time to produce software that could potentially be shipped. While early on the software may not be a fully shippable product, it is a prototype. Upon completing development, user acceptance testing can be used to capture feedback to iterate on in future sprints by creating follow-up stories. Those follow up stories are performing the Iteration phase of human centered design.

Finally, HCD Implementation is reached once a feature is being used by actual workers to perform their tasks. The feature is considered complete and may be shipped out to workers once all of the user stories are completed. It is also important to note that these Agile steps may happen multiple times throughout one or more sprints which also captures the spirit of HCD Iteration as each sprint builds on the prototype until the software is complete.



How Macedon Uses HCD

At Macedon, we have used human centered design to help drive the success of our projects. As an example, one of our clients had their workers on the factory floor, moving and packaging a variety of different products at large scale. Product movement was tracked through pen and paper, and then manually transferred into a computer system. Client managers wanted to streamline and simplify the process for their multilingual workers, leveraging mobile scan guns.

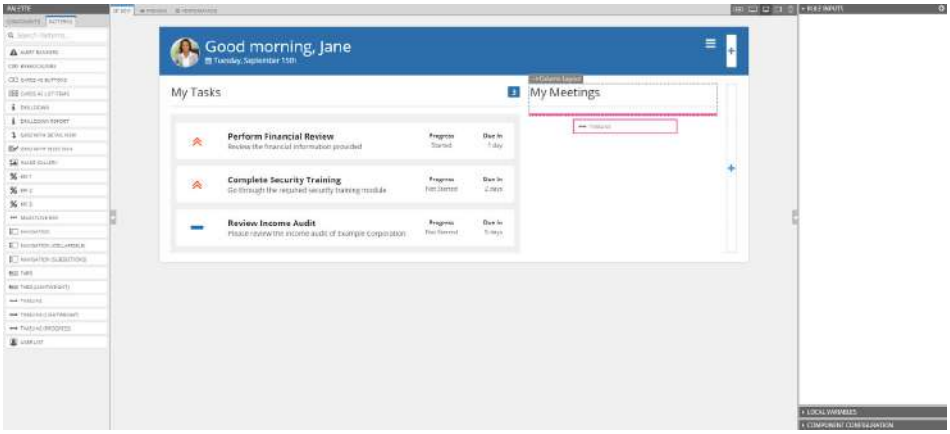
We talked with the workers about their needs and found that their legacy tracking system used different shape and color stickers to track all of their

different products so that each item could be identified easily. We added those familiar symbols into our software so that the workers could easily find the products they wanted to log without having to read a lot of text. Similarly, their legacy system included a big whiteboard that managers used to manually track order statuses. Included with our solution was a large TV screen where workers could scroll through current orders and associated statuses, obviating the need to manually copy this information onto the board.

This made it much easier for the workers to see tasks in progress, increasing productivity by 40%. The solution was adopted by the entire warehouse staff with plans to roll it out to other locations.

Macedon's use of the Appian platform enables us to perform some HCD activities much more effectively. For example, in non-Appian projects, creating prototype interfaces from scratch can be time consuming, which severely reduces scalability when many iterations are needed to find a design that most resonates with the stakeholders. Appian's low code interface designer is a great solution to this problem.

Appian's drag-and-drop functionality allows for rapid prototyping of functional user interfaces in a fraction of the time it takes in a traditional development project. Appian offers a palette of pre-built, styled, and easily customizable interface components that are ready to be dropped onto an interface. Due to our mastery of the interface designer and the SAIL language behind it, Macedon consultants use live mockups of even the most complex interfaces to minimize confusion, get the best feedback, and promptly address that feedback.



It is important to remember that prototyping should not just include UIs but also processes. Appian's process modeler speeds up the design of functioning processes by taking care of the low level details that would have to be implemented in typical programming languages, allowing the developers to focus on the higher level workflow. Macedon's consultants are highly skilled in Appian and its powerful process designer, and can quickly design process iterations for validation or further refinement.

While the powerful Appian interface designer and process modeler allow for faster iterations, time and value of the feedback still need to be kept in mind

throughout the software development lifecycle. Macedon's Project Leads have mastered these tradeoffs and manage scope properly.

Conclusion

HCD leads to software that allows workers to perform their tasks efficiently, empowers individuals with disabilities, and provides a solid return on investment if implemented correctly. It puts workers in control of the software that they ultimately will be using.

Using HCD in tandem with Appian and the Agile software development methodology establishes a strong path to success. Appian provides a fantastic set of tools that facilitates the six phases of HCD, giving development teams the ability to quickly iterate through multiple designs and create software that users will both love and appreciate. The Agile software development methodology further adds to the mix by lining up well with the stages of HCD. Unlike other software development methodologies Agile is flexible which is essential for HCD because being able to iterate rapidly and address feedback is essential. While Agile is backing the phases of HCD, Appian is enabling efficient application of the phases of HCD. Used in tandem, they establish a strong path to success.

Macedon's consultants are highly skilled in Appian and use Agile software development methodology on a regular basis. We have obtained measurable success on projects using human centered design. This makes us an excellent choice when looking to apply a human centered design approach backed by Appian and the Agile software development methodology to software development.

About the Authors

Liviu Catuneanu Cica is a Consultant at Macedon Technologies. He has a passion for UI/UX design and has brought his skill and understanding of the subjects to multiple projects.

Neil Kenney is a Consultant at Macedon Technologies. His past experience involves designing and building accessible software systems.

[1] *IDEO's human centered design process: How to make things people love* [Internet]. UserTesting. 2018.