Design centrado no usuário no desenvolvimento de software

Bruno Sesso

Supervisor: Maria Laura Martinez

Instituto de Matemática e Estatística
Universidade de São Paulo

Bacharelado em Ciência da Computação

São Paulo
2018

Resumo

Em um mundo fictício, onde programação acontece magicamente, aspectos técnicos do desenvolvimento de software, como arquitetura de código, algoritmos, integração com servidores e bancos de dados, etc., passam a ser irrelevantes. Imaginando que um mundo como esse seja real, uma pessoa poderia vir a pensar que não resta mais nada a ser feito para se desenvolver um novo software, ou que só resta idealizar a sua aparência. Entretanto, além do aspecto visual, também é necessário planejar o comportamento do software, ou seja, que mudanças ocorrerão quando alguém clicar um botão, e eventualmente que sentimentos e emoções surgirão com o uso do sistema. Mais do que isso, é necessário projetar o que o software será, isto é, quais problemas e de qual forma ele os resolverá. No mundo real é claro que tal cenário fictício é impossível, mas com esse exemplo, fica claro a dimensão dos aspectos de design, muitas vezes esquecidos como parte do processo de desenvolvimento de software. O design centrado no usuário, em inglês *user-centered design* (UCD), é um método que em busca de planejar tais aspectos, procura entender os problemas enfrentados pelos usuários e a melhor forma que o software sendo criado pode ajudá-lo. Nesse trabalho, procura-se compreender melhor o que é o design centrado no usuário e como ele está presente na área da computação.

Índice

Introdução

O problema de desenvolver um produto pensando em seu usuário final não é um problema exclusivo da área de computação. Imagine, por exemplo, uma panela que tem um cabo de ferro. Certamente o projetista não pensou no conforto de quem iria utilizar esse produto e assim, criou um utensílio que provavelmente irá queimar a mão de quem usá-lo. Em outras áreas como construção civil, produção industrial, produção gráfica, etc. é comum haver um profissional, tal como o arquiteto, designer industrial, designer gráfico, etc., que se preocupa com aspectos do conforto de quem irá utilizar o produto, não somente por um ponto de vista centrado na tecnologia, mas também sob aspectos humanos, sociais e na forma em que o produto encontra o usuário final. No entanto na recente área da produção de software, ainda não existe um profissional totalmente especializado nessa tarefa, e portanto profissionais sem o devido treinamento acabam sendo encarregados de projetar um software.

É muito comum encontrar casos nos quais determinados softwares não atendem as necessidades de quem o usa; softwares que requerem que pessoas entendam o funcionamento e a lógica do computador por trás de sua interface. Há décadas atrás, quando os computadores eram máquinas imensas utilizadas somente por equipes altamente técnicas, esse problema não era tão evidenciado. No entanto, com o surgimento do computador pessoal e a intensificação do seu uso no ambiente de trabalho, o público alvo de computadores mudou completamente para um público sem conhecimento algum sobre seu funcionamento. Isso se intensificou ainda mais durante os últimos anos com o barateamento de eletrônicos, tornando computadores cada vez mais acessíveis, e com a diversificação de produtos, tais como notebooks, tablets, smartphones, etc. Tem-se um público imenso de usuários e passa-se a não ser viável que todos tenham um conhecimento técnico para poder utilizar esses produtos. Um produto que exija menor adaptação do cliente também tem maiores chances de ter sucesso e além disso, faz com que o produto atinja um maior grau de qualidade. Para chegar a esse objetivo, é necessário pensar nas capacidades e necessidades do usuário durante o desenvolvimento. Assim como na construção civil o arquiteto se preocupa com os aspectos humanos e como as pessoas irão interagir com o ambiente, no desenvolvimento de software também há a necessidade de alguém responsável por considerar o usuário e sua interação com o sistema.

Motivação

Imagine um cenário no qual uma pessoa está muito frustrada, pois está utilizando um software para completar uma tarefa, mas ela não consegue realizá-la, porque o software

não funciona como ela imagina. Ou então o cenário onde um novo software é adquirido por uma empresa para otimizar uma tarefa, mas não chega a ser utilizado, pois o software é tão complexo que acaba sendo descartado e a tarefa volta a ser realizada manualmente. Tais cenários não são difíceis de serem imaginados, pois são bastante comuns na atualidade. Desenvolvedores de software são geralmente pessoas altamente técnicas e com alto conhecimento sobre o funcionamento de programação e computadores. Quando decisões de design são tomadas por tais profissionais, há uma tendência à concepção de produtos digitais que exijam que o usuário pense como um computador para conseguir utilizá-lo. Isso tende a ser um problema, pois a grande maioria dos usuários não tem um alto conhecimento sobre o funcionamento de computadores e por conta disso, o produto criado pode acabar não tendo utilidade alguma para eles. Para ajudar a solucionar esse problema o design centrado no usuário coloca a pessoa que irá utilizar um produto no foco do seu processo de criação. Dessa forma acredita-se que o produto criado estará mais próximo do que o usuário quer e necessita.

O design centrado no usuário é uma das soluções utilizadas atualmente para evitar esse tipo de deficiência. Busca-se entender qual os problemas enfrentados pelo usuário e como o software pode suprir suas necessidades para só então, explorar inúmeras possíveis soluções e enfim chegar em uma que possivelmente será criativa e inovadora.

Justificativa

Muitas vezes cursos de computação abordam técnicas de engenharia de software, algoritmos, matemática, etc., porém pouca atenção é dada para técnicas de design de um software. Por design de software, não estamos nos referindo somente ao aspecto visual de um sistema, mas ao grande processo que transforma uma ideia ou problema inicial em um conceito de software e só então passa a ser codificado no software final. Esse processo pode envolver o desenvolvimento do aspecto visual como também o comportamental e emocional, além de envolver técnicas, ferramentas e atividades durante sua aplicação. Criar programas de qualidade que atendam as necessidades de quem o usa e a forma de como se fazer isso é uma grande preocupação no desenvolvimento de software. Portanto, estudar o campo do design centrado no usuário e como ele tangencia a área da computação se mostra uma tarefa de grande interesse e será abordada nesse trabalho.

Objetivos

Compreender os seguintes itens:

- O que é o design centrado no usuário.
- De onde vem o design centrado no usuário.

- O porquê da popularidade do design centrado no usuário na atualidade.
- A diferença de design centrado no usuário e termos semelhantes como interação humano-computador, design de interação, experiência de usuário e design de interface.
- Como funciona o design centrado no usuário.
- O papel do design centrado no usuário na computação.
- Como o design centrado no usuário é utilizado no desenvolvimento de software.

Estrutura

Esse trabalho foi dividido em 4 seções, nas quais são discutidas informações que ajudarão a cumprir cada um dos objetivos.

1. Conceituação

A discussão de termos comumente confundidos é feita de forma breve.

2. Contextualização Histórica

Busca-se compreender as origens de UCD através do estudo da história de Interação Humano Computador, para então refletir sobre sua relevância na atualidade. Estudar a história também ajudará a compreender o que de fato é UCD.

3. Design Centrado no Usuário

Um estudo mais completo sobre como funciona o design centrado no usuário é feito. Aspectos de sua aplicação são discutidos.

4. Design Centrado no Usuário e Desenvolvimento de Software

De que forma o design centrado no usuário se integra no processo de desenvolvimento de software.

---

1. Conceituação

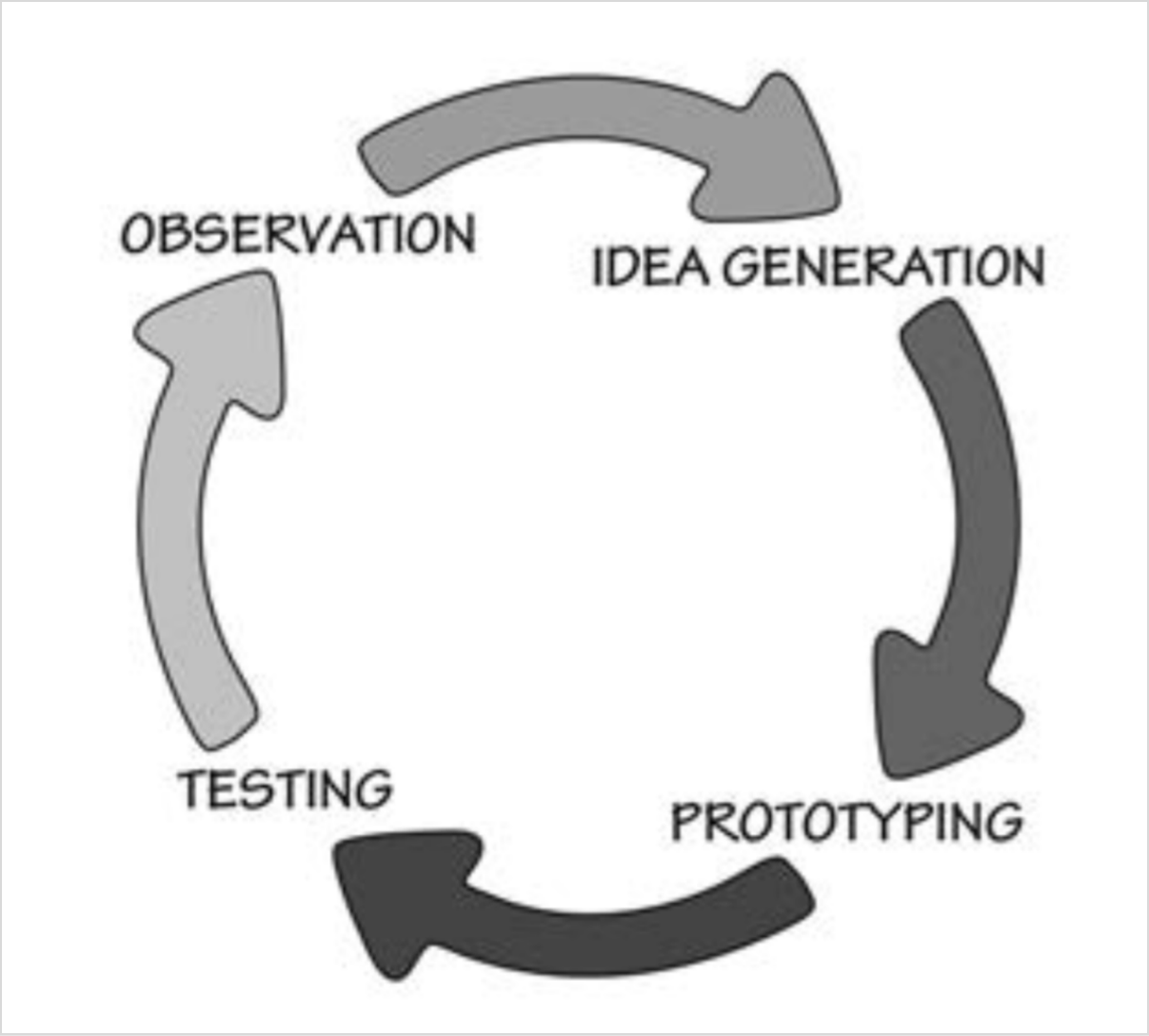
Durante a pesquisa de temas relacionados ao design centrado no usuário é comum encontrarmos casos onde diferentes termos acabam sendo utilizado em situações semelhantes e podem ser erroneamente tomados como sinônimos. Para entender melhor o escopo do design centrado no usuário, buscamos compreender as diferenças dos seguintes termos:

- Design centrado no usuário
- Design de interação
- Experiência de usuário
- Interface de usuário
- Interação humano computador

Tais termos não tem sua definição extremamente bem definida com informações precisas sobre seu escopo. Frequentemente são moldados pelo seu uso ao longo do tempo e podem ter um significado, de certa forma, vago. Há muita discussão sobre diferenças minuciosas e o que cada termo compreende. Nesta seção buscaremos apresentar o significado de cada termo, sem nos atentar a detalhes muito profundos, de forma a prover informação suficiente para que o leitor seja capaz de distingui-los.

Design Centrado no Usuário

Em inglês *User-Centered Design* (UCD) é um termo firmado por Norman & Draper (1986 apud Ritter et al., 2014).
UCD coloca o usuário e suas necessidades no centro de todas as decisões de design a serem feitas. Isso é feito através de um processo iterativo frequentemente caracterizado por 4 etapas:



(Norman, 2013)

Durante a etapa de observação busca-se entender o contexto de uso do produto sendo desenvolvido através de entrevistas, enquetes, observação, etc. Durante a fase de geração de ideia busca-se expandir inúmeras soluções, geralmente fazendo um brainstorming, e a partir de então tenta-se chegar a um conceito. Na fase de prototipação busca-se construir um protótipo rápido e barato do conceito elaborado na etapa anterior para ser testado e avaliado durante a fase de teste. Muitas vezes isso é feito através da observação do uso do protótipo por algum possível usuário (Norman, 2013). Frequentemente o processo de UCD pode ser apresentado por um outro conjunto de etapas, mas que ainda assim consiste das mesmas atividades agrupadas de outras formas.

Design de Interação

Em inglês *Interaction Design* (IxD), é uma disciplina voltada para a área de design bem mais do que para engenharia e ciência (Cooper, Reimann & Cronin, 2007). Essa disciplina busca estudar o comportamento da tecnologia e como as pessoas interagem com ela. Outros profissionais de design, ao tentar fazer o design de produtos digitais,

frequentemente se focavam na forma estática e não na interatividade que essa tecnologia proporciona. Produtos digitais apresentam um comportamento altamente complexo, pois frequentemente mudam de acordo com a iteração com o usuário. Ao clicar em um botão, a interface pode vir a mudar, um processo pode ser executado, e muitas outras respostas que não são comuns a produtos não computacionais podem ocorrer. Fazer bom uso dessas interações de forma que o usuário as entenda é tarefa de IxD. O design de interação faz uso de outras disciplinas como psicologia, design, arte e emoção para criar uma experiência agradável e positiva. Dessa forma ao utilizar uma tecnologia, o usuário deve ser capaz de entender o que pode ser feito, o que está acontecendo e o que acabou de acontecer (Norman, 2013).

Experiência de usuário

Em inglês *User Experience* (UX), segundo a ISO 9241-210 é:

"A person's perceptions and responses resulting from the use and / or anticipated use of a product, system or service."

Ela está relacionada a todas as reações em uma pessoa, geradas pelo uso de um produto, isto é, suas emoções, reações físicas e fisiológicas, etc. Por exemplo um fone de ouvido pode ter uma péssima qualidade de som e uma péssima aparência, mas ao mesmo tempo trazer uma boa experiência à uma pessoa que seja fã do jogador de basquete que fez o comercial desse produto. Ou seja, UX não envolve somente o produto em si, mas também os outros recursos em sua volta como propagandas, embalagem, etc. A área de design de experiência (*experience design* ou *UX design*) se preocupa com os sentimentos e emoções das pessoas e busca projetar produtos, processos, serviços, eventos e ambientes focados na qualidade e satisfação da experiência total (Norman, 2013).

Interface de usuário

Em inglês *User Interface* (UI), é o espaço onde ocorre a interação entre o usuário e o produto. Frequentemente é associado às GUIs (graphical user interfaces) onde a maior parte da interação ocorre, mas também pode se referir à interfaces de audio, dispositivos físicos, etc. O design de interfaces de usuário (UI design) foca, frequentemente, no design da aparência e estilo de uma interface.

Interação Humano Computador

Interação Humano Computador (IHC) é também conhecida como, interação humano máquina, ou em inglês como *human-computer interaction* (HCI).
IHC é uma disciplina que, como o nome diz, estuda *interações* entre *humanos* e *computadores*. Enquanto seres humanos, se comunicam com outras pessoas através de

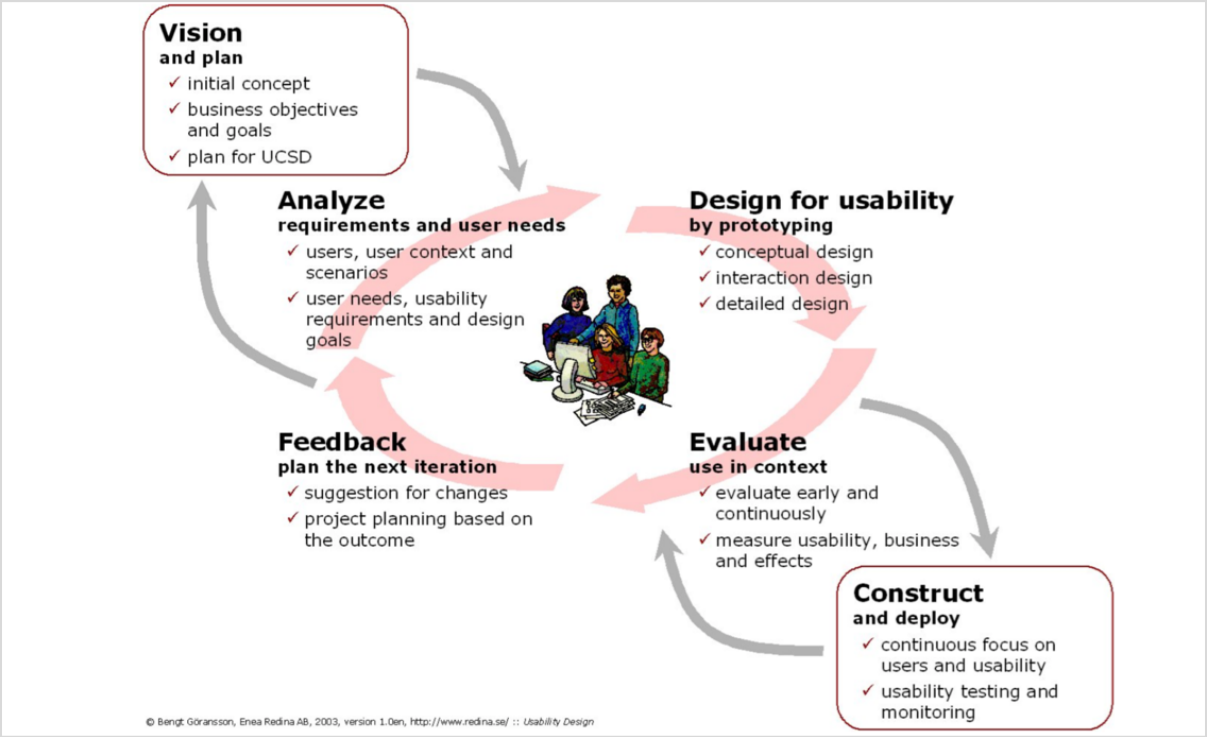
> voz, texto, gestos corporais, expressões, etc., computadores realizam essa mesma tarefa através de troca de bits, isto é, sinais elétricos. Como a forma de comunicação de ambos é diferente, em algum momento deve haver algum tipo de "tradução", que geralmente é feita através de interfaces tais como: teclado, mouse, tela, etc. A área de IHC busca entender, avaliar e aprimorar formas da interação entre o homem e a máquina e é bastante relacionada à área de engenharia de usabilidade, isto é, o estudo da criação de softwares e outras tecnologias as quais pessoas iram querer usar, conseguir usar e achar eficiente quando usá-las. IHC é uma área extremamente multidisciplinar, fazendo uso de disciplinas como: psicologia, ciência da computação, design, comunicação, ciências sociais, linguistica, ergonomia, e muitas outras. São campos de pesquisa de IHC: métodos para design de interfaces, avaliação de interfaces, estudo de implicações sócio-culturais, entre outros.

Boa parte da confusão entre esses termos ocorre pelo fato dos termos terem a palavra design associada. IxD se refere ao design de interações, UX design ao design de experiências e UI design ao design de interfaces. No entanto o termo *design centrado no usuário* tem um significado um pouco diferente. UCD não significa o design da *centricidade* do usuário, mas sim o processo de design cujo centro é o usuário. Durante a fase de design desse processo pode-se dar foco ao design de interação, experiência, interface de usuário, etc. Essas áreas muitas vezes se sobrepõe de forma a se complementar: aspectos visuais podem vir a ser importantes em UX, que por sua vez podem ser relevantes ao se fazer o design de interação, etc. Porém, o foco de cada uma continua a ser diferente.

---

2. Contextualização Histórica

De acordo com Norman (2013), o design centrado no usuário é o processo que garante que o produto condiga com as necessidades e capacidades das pessoas para as quais ele foi projetado. Através do foco em entender o usuário e suas necessidades, esse processo passa por algumas etapas características para desenvolver um produto que melhor atenderá quem o usa:



Gulliksen et al. (2003)

Para compreender a importância do design centrado no usuário e a sua relevância no cenário atual é interessante estudar a sua origem e o contexto histórico no qual ele se encontra, principalmente dentro da área da computação.

O termo *design centrado no usuário* foi firmado pelo famoso pesquisador Donald A. Norman e se popularizou principalmente após a publicação do livro *User-Centered System Design: New Perspectives on Human-Computer Interaction* (Norman et al. 1986 apud Ritter, Baxter & Churchill, 2014). No entanto essa área não surgiu de repente e nem dependeu de somente um evento para ser desencadeada. Ela foi resultado de um complexo contexto onde se encontrava a tecnologia e as pesquisas de diversas disciplinas, tais como a engenharia de software, psicologia e sociologia. Para entender esse contexto, será estudada a origem do design centrado no usuário através da história da interação humano computador.

Durante a década de 1960, avanços na tecnologia de hardware resultaram em um aumento da capacidade computacional dos computadores, que em conjunto com o crescimento da demanda e do tamanho dos softwares acarretou no que se chamou de *crise do software*. Os maiores sintomas dessa crise foram custos acima do previsto, atraso nas entregas, softwares instáveis e inefetivos, e códigos difíceis de se manter. Em frente a essa crise, notou-se a necessidade em melhorar o processo do desenvolvimento de software e deu-se origem ao campo de *engenharia de software*. De acordo com Caroll (2001), diante de tais acontecimentos, quatro cenários independentes passam a ocorrer simultaneamente, dando estrutura para o nascimento do campo de IHC na década de 1980:

1. Prototipação e desenvolvimento iterativo

Uma das soluções apresentadas por estudos iniciais do novo campo de engenharia de software foi o modelo em cascata, que divide o desenvolvimento de software em etapas distintas, sequenciais e de sentido único.

Implicações:

Notou-se que requisitos críticos tendem a surgir durante o desenvolvimento do sistema, e o modelo em cascata não permite que etapas sejam retrocedidas. Isso incentivou modelos de desenvolvimento de software que são iterativos, onde tenta-se chegar a uma solução parcial para um sub-conjunto de requisitos. A partir de então faz-se o uso de protótipos para descobrir novos requisitos e possivelmente reformular os objetivos.

2. Ergonomia e Software Psychology

Durante a década de 1970 o ato de programar passou a ser objeto de estudo da psicologia. Abordagens behavioristas passam a tentar entender o ato de programar, desenvolver software e como as pessoas interagiam com sistemas interativos.

Implicações:

Novos artifícios que auxiliam programação são estudados e implementados. Por exemplo: variáveis com nome, comentários *in-line*, fluxogramas, etc.
Estudos de como as pessoas utilizam computadores: tempo de resposta, como pessoas especificam um comando para o computador, etc.
Produtores de computadores passaram a estabelecer laboratórios de usabilidade.

3. Interfaces de usuário

Durante a década de 1970 graças aos avanços dos computadores e displays, estudos e metáforas da interface passaram a se desenvolver rapidamente.

Implicações:
Muitas das metáforas que são usadas até hoje foram desenvolvidas. Por exemplo a metáfora da área de trabalho, ícones, barras, menus, etc.

4. Modelos e teorias

Na década de 1970 área de ciência cognitiva (estudo da mente humana) passa a usar a computação como área de aplicação para suas teorias, fazendo estudos a respeito de princípios da percepção, atividade motora, comunicação, linguagem, comportamento de grupos, etc.

Implicações:
Desenvolvimento de teorias iniciais da área de IHC, por exemplo: *GOMS (Goals, Operators, Methods, Selection rules)*

Apesar de terem suas origens em diferentes áreas do conhecimento, essas quatro linhas separadas se encontram na década de 1980 e ajudam a consolidar a área de Interação Humano Computador através da formação de diversos grupos de estudo, incluindo o grupo da ACM: SIGCHI (Special Interest Group in Computer-Human Interaction) em 1982. Nesse primeiro momento as pesquisas se centravam em dois principais temas: software e métodos.

Foco em software

Foco em inventar e refinar conceitos de interface de usuário e técnicas para fazer um sistema melhor com usabilidade e utilidade aprimorada. Conceitos de interface de usuário, metáforas, técnicas de interação e métodos de desenvolvimento de software são objetos de estudo dessa área. Vale ressaltar que não restrito a interfaces que são mais comuns durante o atual uso de computadores, pesquisas em técnicas e métodos mais apropriados para outros tipos de interface como voz, gestos, sensores de posição, etc. também passaram a serem estudadas.

Com o aprofundamento dessas pesquisas, conferências como a *User Interface Software & Tools* da ACM surgiram, indicando que uma quantidade considerável de pesquisadores da área de IHC passaram a adotar esse foco.

Foco em métodos

Foco em pesquisas empíricas, modelos e teorias sobre usabilidade. Através de estudos de pessoas utilizando sistemas e formas de se avaliar tais sistemas, a área conhecida como engenharia de usabilidade passou a se formar. Ao passar do tempo subcomunidades dentro da engenharia de usabilidade passaram a surgir com diferentes pontos de vistas e

focos de pesquisa: Conference on User Modeling (desde 1986), Usability Professionals Association (UPA) (desde 1991, atual User Experience Professionals Association, UXPA).

Sociólogos e antropólogos passaram a estudar a interação com o sistema de forma mais ampla, onde o uso de sistemas envolve o ambiente e o grupo de pessoas no seu entorno, principalmente através de estudos sobre o ambiente de trabalho e ferramentas colaborativas. Antecipar as ações de usuários e considerar o ambiente e grupos em modelos e teorias passa a ser uma tarefa bastante difícil. Há um movimento para se incluir o usuário durante o processo de desenvolvimento do produto e isso acaba acarretando no surgimento de outras subcomunidades: Participatory Design Conference (desde 1990), Conference on Computer-Supported Cooperative Work (desde 1986)

Visando integrar os variados estudos do foco de métodos com o de software, surge o *User-centered System Design* que futuramente passou a ser chamada apenas por *User Centered Design* (design centrado no usuário), termo firmado por Norman et al. (1986 apud Ritter et al., 2014), no qual os autores ressaltavam a importância em ter um design guiado pelo entendimento do usuário e suas necessidades:

"But user-centred design emphasizes that the purpose of the system is to serve the user, not to use a specific technology, not to be an elegant piece of programming. The needs of the users should dominate the design of the interface, and the needs of the interface should dominate the design of the rest of the system." (Norman et al., 1986)

Pode se dizer que as quatro etapas a seguir formam o esqueleto do processo do design centrado no usuário e é fácil notar como, de fato, esse processo tenta integrar os diversos focos de pesquisa de IHC do momento:

1. Resultados dos campos de estudo apresentados anteriormente, tais como a observação do uso do sistema em contexto, participação do usuário durante o desenvolvimento, entrevistas com stakeholders, etc. são essenciais para ter um entendimento do usuário, suas necessidades e das restrições do projetos. São portanto etapas essenciais durante a fase inicial de planejamento e pesquisa, antes do design do produto em si.

2. O design do produto usa como base as informações adquiridas através da pesquisa da etapa anterior em conjunto com guidelines de design, providas por estudos de ergonomia e ciência cognitiva, e também conceitos, metáforas e técnicas estudadas pelo foco em software de IHC.

3. Protótipos são construídos e avaliados através de interações com usuários e métodos de

avaliação estudados pela engenharia de usabilidade.

4. Novas propostas e ideias podem vir a surgir, e a partir dos resultados coletados pode-se ter uma nova perspectiva sobre o produto a ser desenvolvido. Ocorre, então, a iteração e volta-se para o estágio inicial de pesquisa e ideação (1).

Graças ao invento da internet, a partir da década 1990 computadores passam a ser utilizados por um número cada vez maior de pessoas e assim, passa-se a haver um novo foco de uso: a comunicação. A princípio, o principal objetivo da internet era auxiliar a realização de computação remota, porém notou-se que a maior parte dos seus usuários estavam utilizando-a para se comunicar através de e-mails, grupos, chats, etc. (Winograd, 1997). Além disso avanços tecnológicos tornaram computadores mais acessíveis e ajudaram a popularizar outros dispositivos como tablets, smartphones, etc. Esse ambiente onde a informação atinge um número enorme de pessoas se mostrou a estrutura perfeita para se gerar lucros por meio do e-commerce, anúncios, e outros serviços. O impulso gerado pela busca do lucro e a alta competição, consequente do crescente uso da internet, coloca em questão preocupações do marketing como a captura de olhares, experiência do usuário e usabilidade, trazendo para o mundo da computação profissionais como o designer. O olhar do designer sobre aspectos estético-visuais e sobre a satisfação do usuário proporciona novas perspectivas à área de IHC, que até então buscava a criação de interfaces somente através da ciência e engenharia (Grudin, 2012). Como consequência disso, em 1995 deu-se início à conferência da SIGCHI: *Designing Interactive Systems (DIS)* que apesar de seu propósito, acabou não atraindo muitos designers visuais. Somente em 2003, a SIGCHI em conjunto com a ACM SIGGRAPH (*Special Interest Group on Graphics and Interactive Techniques*) e a AIGA (*American institute of Graphic Arts*) iniciaram a conferência *Designing for User Experience (DUX)*. Ainda que essa série de conferências só tenha durado até 2007, ela foi suficiente para trazer muitos designers visuais e comerciais para a computação e para mostrar a importância do design até mesmo no ambiente científico de IHC.

Em consequência dos avanços científicos de até então, da atual tecnologia e do contexto social em que vivemos, assuntos muito discutidos dentro da área de IHC atualmente incluem: experiência do usuário, influência das emoções humanas, computação ubíqua, computação social, entre outros. Isso não torna obsoleto pesquisas de temas estudados no passado como conceitos de interfaces, modelos e teorias de usabilidade, etc.

Discussão

É interessante notar como as disciplinas e focos de estudos refletem diretamente o estado

de evolução tecnológica do computador e o contexto social em que ele se encontrava. A princípio, computadores eram máquinas imensas, que apesar do seu grande potencial para a ciência, eram extremamente difíceis de serem utilizadas. Isso acabou chamando atenção de pesquisadores que tentavam compreender o funcionamento da mente humana para melhorar a interação com computadores. Em um cenário mais a frente na história, a tecnologia evoluiu permitindo que computadores passassem a ser utilizados em escritórios por pessoas sem muito treinamento. Nesse momento, o grande aumento do número de usuários e o contexto social do ambiente de trabalho passaram a chamar atenção de sociólogos e outros profissionais que passaram a estudar a interação do humano com o computador sobre um novo ponto de vista.

Durante a década de 1990, com o surgimento do comércio eletrônico e todo um mercado que girava em torno da internet, passa-se a haver uma maior preocupação em *agradar o usuário* e assuntos como usabilidade, experiência do usuário, design de interação, etc. começam a ganhar espaço. Nesse momento, diversos autores descrevem a falta de softwares que atendam as necessidades de seus usuários e a necessidade de profissionais especializados no design de um software. Esses autores criticam o fato de que na maior parte dos casos softwares eram projetados pelos próprios programadores. Logo esses profissionais seriam responsáveis pelo design do comportamento do software e pela forma que os usuários teriam que interagir com ele. Como na maior parte dos casos, programadores não tem um treinamento adequado sobre como projetar um software que cumpra seu objetivo e que seja agradável ao usuário, muitos dos softwares criados eram frustrantes e difíceis de serem usados:

> *"In the next fifty years, the increasing importance of designing spaces for human communication and interaction will lead to expansion in those aspects of computing that are focused on people, rather than machinery. The methods, skills, and techniques concerning these human aspects are generally foreign to those of mainstream computer science, and it is likely that they will detach (at least partially) from their historical roots to create a new field of 'interaction design'."*
> *(Winograd, 1997)*

> *"The most important social evolution within the computing professions would be to create a role for the software designer as a champion of the user experience…. What is design? … It's where you stand with a foot in two worlds—the world of technology and the world of people and human purposes—and you try to bring the two together.*
> *(…)*
> *Today, the software designer leads a guerrilla existence, formally unrecognized and often unappreciated. There's no spot on the corporate organization chart or career ladder for*

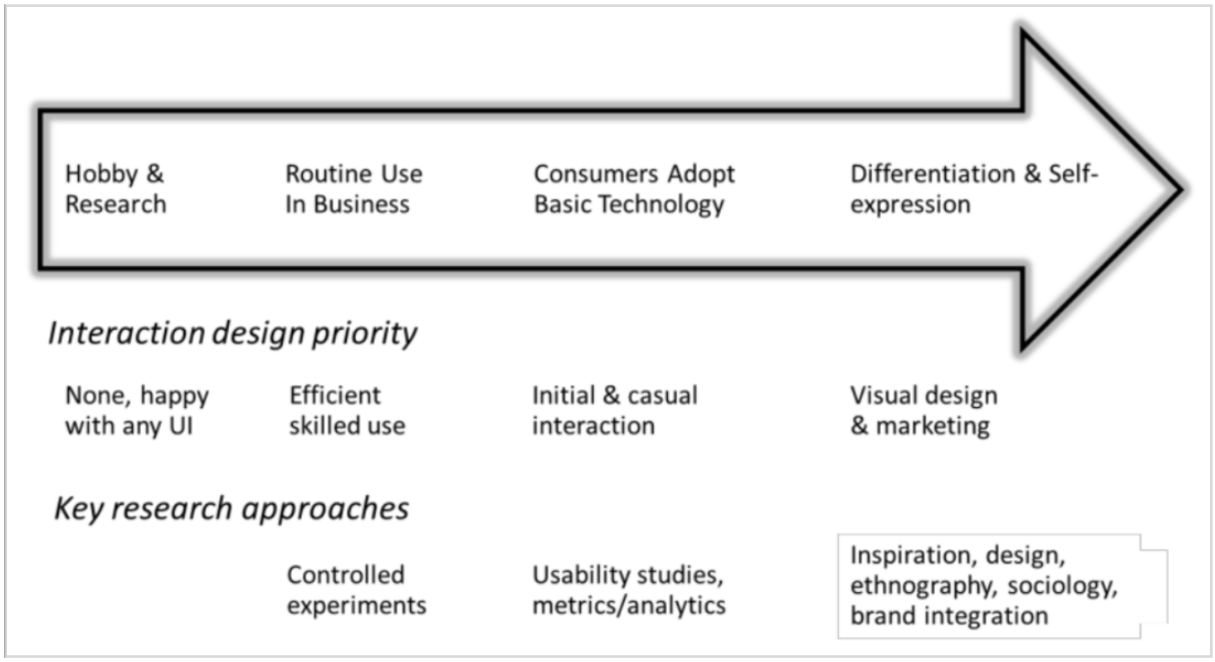
> *such an individual. Yet time after time I've found people in software development companies who recognize themselves as software designers, even though their employers and colleagues don't yet accord them the professional recognition they seek.*
> *Design is widely regarded by computer scientists as being a proper subpart of computer science itself. Also, engineers would claim design for their own. I would claim that software design needs to be recognized as a profession in its own right, a disciplinary peer to computer science and software engineering, a first-class member of the family of computing disciplines."*
> *(Kapor, 1990)*

> *"At the time About Face was first published in August 1995, the landscape of interaction design was still a frontier wilderness. A small cadre of people brave enough to hold the title user interface designer operated under the shadow of software engineering, rather like the tiny, quick-witted mammals that scrambled under the shadows of hulking tyrannosaurs. "Software design," as the first edition of About Face referred to it, was poorly understood and underappreciated, and, when it was practiced at all, it was usually practiced by programmers. A handful of uneasy technical writers, trainers, and product support people, along with a rising number of practitioners from another nascent field—usability—realized that something needed to change."*
> *(Cooper et al., 2007)*

Uma metáfora bastante comum é a do arquiteto e o engenheiro civil. Durante a construção de uma casa esses dois profissionais são essenciais. O engenheiro tem um conhecimento sobre as tecnologias necessárias para de fato realizar a construção; como construir um muro, calcular a posição das vigas, garantir que a casa não irá cair, etc. Ele é capaz de projetar uma casa, seus espaços, posicionamento, etc. No entanto é o arquiteto quem irá melhor pensar como os moradores da casa irão interagir com ela, como os ambientes devem interagir entre si, que tipo de iluminação é adequada, etc. Só então o arquiteto passará a projetar o edifício e seus espaços. Da mesma forma, engenheiros da computação, programadores e outros profissionais dessa área tendem a ter um alto conhecimento tecnológico do desenvolvimento de software. Assim como engenheiros civis eles sabem como construir as *paredes* de um software. No entanto, na produção de software nota-se a falta de um profissional como o arquiteto que irá levar em consideração a interação do usuário com o produto para só então, fazer o design do software.

Apesar de mais de uma década ter se passado desde que os autores citados acima tenham relatado a necessidade por esse novo tipo de profissional, ainda não parece haver um consenso sobre o título desse tipo de trabalho e sobre seu papel. Muitas empresas

oferecem cargos como *user experience designer*, *experience designer*, *interaction designer*, *user interface designer*, *information architect*, entre outros. No entanto em diferentes empresas o mesmo título pode ter papéis diferentes, ou diferentes títulos terem os mesmos papéis. Termos como usabilidade, experiência de usuário, design centrado no usuário e design de interação estão cada vez mais presentes tanto na comunidade de computação como em muitas outras áreas como o design, administração, marketing, etc. Muitos autores concordam que as fronteiras entre o significado desses termos é meio incerta e até o mesmo termo pode vir a ter diferentes nuanças quando tratado por pesquisadores de diferentes áreas. Ainda assim, tais termos tem suas diferenças e não devem ser tratados como sinônimos como vêm sendo feito por uma boa parcela da comunidade de desenvolvimento de software. Por outro lado, a popularização desses termos mostra uma crescente preocupação em melhor considerar o usuário durante o processo de produção de um software. Em Grudin (2012) o autor defende que a popularização dessa preocupação possa estar relacionada com o *caminho de maturação* de muitas tecnologias.



(Grudin, 2012)

Essa imagem reflete o caminho de maturação de diversos tipos de tecnologia, inclusive a do desenvolvimento de software. Uma das principais alterações a se notar é a adaptação do usuário ao computador. Na etapa de hobby e pesquisa o usuário tem sua própria motivação para utilizar a tecnologia e portanto ele se adapta aos desafios de interação encontrados à frente. A introdução da tecnologia em ambiente de trabalho traz um maior número de pessoas menos interessadas em usar a nova tecnologia. Na próxima etapa, o marketing busca trazer mais ainda desses usuários, até então desmotivados, para fazer uso do tal produto inovador. Consequentemente passa-se a haver uma maior

preocupação em trazer o computador para mais perto do homem através da melhoria de interfaces entre ambos. Dessa forma a tecnologia se adapta aos hábitos do usuário mais do que o contrário. Uma pessoa não deve precisar saber que o computador tem um processador, memória, etc. para saber como manuseá-lo.

Esse fenômeno de mudança da tecnologia para melhor atender o usuário ocorre graças ao crescente número de usuários e pelo desejo em expandir o uso da tecnologia em questão, característica típica de empresas que visam o lucro sobre algum produto. Além de usuários, o número de desenvolvedores e empresas de software também é crescente, gerando competição. Atualmente o monopólio de uma empresa sobre um tipo de software é mais raro. Não existe mais somente um tipo de editor de texto, ou somente um cliente de email. Essa competição também abre espaço para melhorias mais rápidas e para a busca da criação de um produto com características diferenciadas. Se uma empresa de software produz um software não tão agradável ao usuário, provavelmente não irá fazer com que seu produto seja muito usado e eventualmente será ultrapassada por algum outro produto concorrente que traga uma melhor experiência. Certamente ainda há um número grande de outros fatores a se levar em conta como a complexidade do software, marketing do produto, público alvo, etc. No entanto a experiência do usuário e o suprimento de suas necessidades não deixam de desempenhar um papel importante no sucesso de um produto.

Resumo

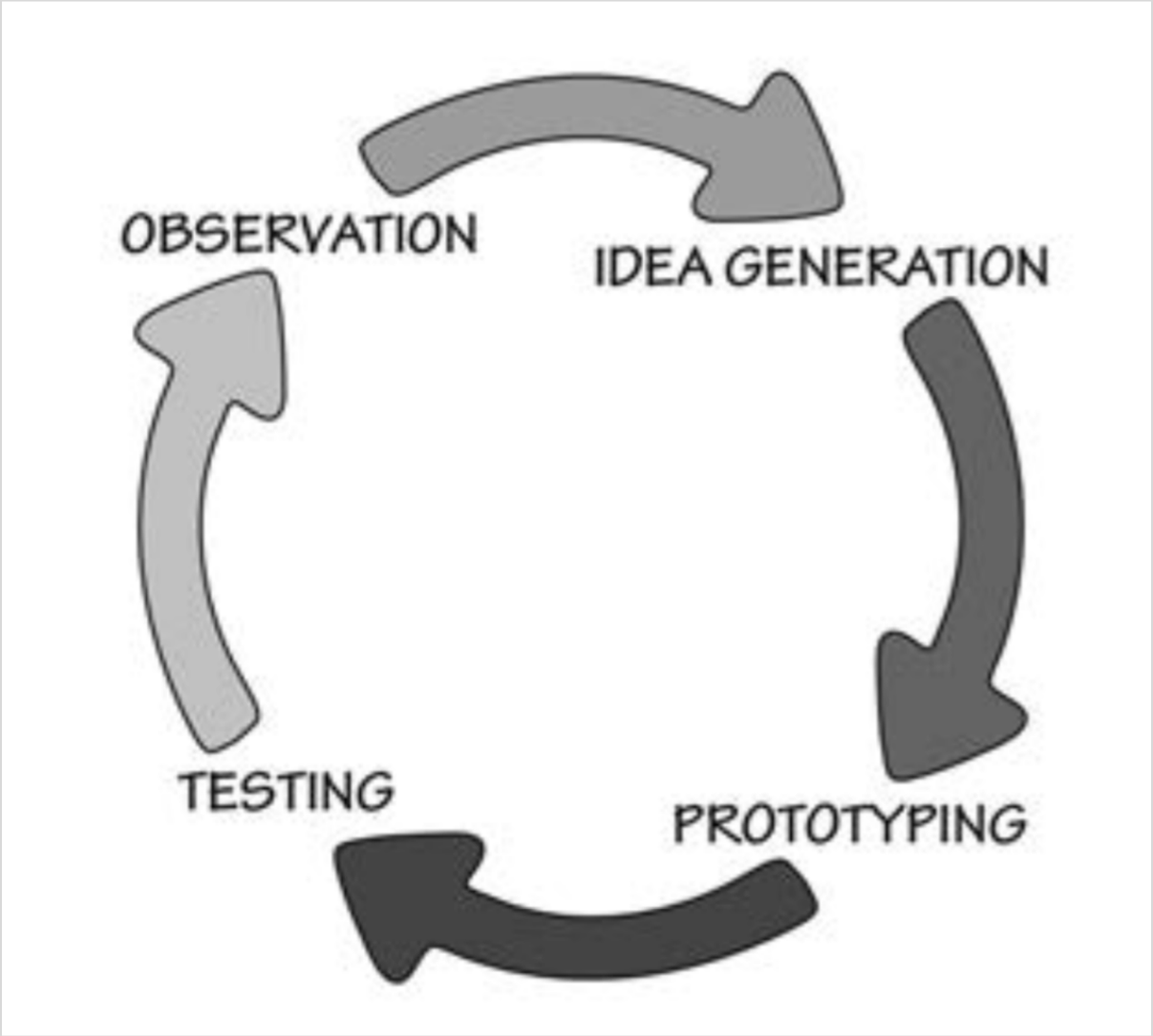
A história da computação ainda é bastante recente, caminhando em um passo acelerado e com muitas mudanças. Naturalmente a melhor das interfaces e como melhor criá-la não surgiu repentinamente junto com o invento do computador. Ao longo da história, notou-se um problema com a interação entre usuários e o computador e a partir de então passou-se a estudar formas de resolver essa questão. Gradativamente soluções foram surgindo, ajudando a trazer novos usuários e a popularizar os atuais diversos tipos de computadores. Percebeu-se que a produção de software não se trata somente do computador, mas também de quem está o usando. A importância da inclusão do usuário no processo de desenvolvimento de software passou a ser mais reconhecida e a partir de então métodos como o design centrado no usuário surgiram. Com o surgimento da internet e a popularização do computador pessoal usabilidade, design visual e experiência de usuário passaram a ser mais discutidos dando mais destaque ao design centrado no usuário. A popularização do design de interfaces e a criação de softwares que melhor atendam as necessidades de seus usuários atraiu diversos profissionais a essa parte do desenvolvimento do software. No entanto parece ainda não haver um consenso sobre qual é o profissional designado para realizar o design de um software e sobre qual

é o escopo de seu trabalho. Por outro lado, vivemos em um momento onde há uma grande preocupação em produzir um software de qualidade que traga uma boa experiência ao usuário.

---

3. Design Centrado no Usuário

O design centrado no usuário é um processo criado para que seja implementado em uma situação real. Ou seja, além de ter o usuário como centro das decisões de design, o processo também considera que eventualmente nem todas as decisões do projeto serão feitas pelo designer e que além dele existem stakeholders e outras equipes. O processo de UCD possui uma característica iterativa e se destaca pelas etapas de pesquisa, design, prototipação e teste. Além disso a prática de UCD é caracterizada por uma série de princípios e filosofias a serem tomadas ao longo do desenvolvimento. A seguir apresentaremos de forma resumida cada etapa. Nosso objetivo é ilustrar o tipo de atividade realizada no processo de UCD e não servir como um guia completo, portanto não nos atentaremos a todos os detalhes e todas as possíveis atividades de cada etapa.



(Norman, 2013)

Pesquisa (Observation)

Durante a etapa de pesquisa busca-se entender quem são usuários.

Além de compreender o contexto de uso do produto sendo desenvolvido e como o usuário irá interagir com ele, procura-se observar as atitudes dos usuários, como eles pensam, valores, motivações, etc. Através de uma série de atividades tenta-se obter informações suficientes para modelar os usuários e os contextos de uso para iniciar a fase de design.

Essa etapa se inicia pela identificação dos stakeholders. Stakeholders são quaisquer pessoas que serão afetadas de alguma forma por qualquer decisão de projeto. Frequentemente stakeholders são usuários, executivos, gerentes, representantes de marketing, etc. e podem ser até mesmo família, amigos ou professores. Provavelmente eles serão os financiadores do projeto e quem decidirão fatores importantes como o orçamento e limite de tempo. Nessa etapa é importante identificar a visão dos stakeholders em relação ao projeto a ser desenvolvido. Frequentemente diferentes departamentos têm diferentes perspectivas sobre o projeto e quais são os reais objetivos. Através de entrevistas e encontros com stakeholders, busca-se compreender e harmonizar esses diferentes pontos de vista. Nesse momento é importante compreender as restrições técnicas e quais oportunidades o produto pode usar para eventualmente vir a desempenhar um papel inovador. Os stakeholders também serão capazes de apontar quais são suas concepções de usuário alvo e informações que serão relevantes para a formulação da pesquisa de usuário. Se necessário, entrevistas com especialistas da área abordada pelo projeto podem ser realizadas para melhor entender detalhes que podem ser relevantes ao design do produto. Esse tipo de entrevista é comum em áreas mais complexas ou mais especializadas.

Entrevistas com usuários são uma atividade muito importante. Eles serão quem de fato utilizarão o produto e são o centro desse processo de design. Através das entrevistas, pode-se conhecer melhor o usuário e suas reais motivações e necessidades. Frequentemente, os próprios usuários não são capazes de analisar seus próprios comportamentos. Portanto, através desse dialogo, busca-se compreender o usuário como ser humano; como ele enxerga seus problemas, o que ele considera bom e o que ele considera ruim, quais são seus objetivos de vida, etc. Além disso, informações cruciais para o design podem ser obtidas, como o contexto de uso do produto, qual o conhecimento dos usuários em como utilizar o produto, quais os tipos de atividades que os usuários realizam que o produto não dá suporte, motivações e objetivos para usar o produto e os problemas com produtos atuais. . Portanto realizar entrevistas dentro do ambiente de uso do produto, ou observar o usuário no contexto de uso pode ser bastante relevante para entender o seu fluxo de trabalho.

Paralelamente com as entrevistas citadas, revisões bibliográficas e análise de aplicações

similares podem ser realizadas. Revisões bibliográficas incluem pesquisas de marketing realizadas até então, análise de dados quantitativos como de enquetes e questionários, notícias e qualquer outro conteúdo que possa se mostrar relevante. Durante a análise de aplicações similares, obtém-se uma ideia do estado da arte e uma ideia geral sobre os pontos fortes e limitações dos produtos disponíveis no momento.

Com os resultados obtidos uma atividade bastante comum é a construção de personas. *Personas* são arquétipos compostos baseados nos dados comportamentais dos usuários entrevistados. Eles representam um conjunto de possíveis usuários baseados em padrões de comportamento observados, isto é, personas funcionam como modelos de usuários. Durante a construção de uma persona os dados obtidos nas entrevistas anteriores são analisados e sintetizados através da identificação de variáveis comportamentais e o agrupamento dos usuários de acordo com elas.

Outra técnica bastante comum é a construção de *cenários* de uso, que podem eventualmente fazer uso das personas. Em um cenário, uma breve narrativa sobre o usuário e como o possível produto se encaixa no complexo contexto de sua vida é feita. Esse tipo de técnica ajuda o designer a compreender melhor o usuário e contribui com a identificação de situações nas quais o produto pode vir a ajudar, levando em consideração características individuais de um usuário. Além de ser uma ferramenta usada para identificar necessidades do usuários ela também é bastante efetiva como forma de comunicar tais necessidades a demais equipes.

UCD valoriza bastante a pesquisa qualitativa sobre a quantitativa. Através da pesquisa qualitativa é possível estudar a fundo as pessoas e dessa forma consegue-se entender melhor os objetivos e necessidades dos usuários. Enquanto a pesquisa quantitativa ajuda a compreender quantidades, isto é, responder *Quanto?*, pesquisas qualitativas ajudam a responder *O que?*, *Por que?* e *Como?* (Cooper et al., 2007). No final dessa etapa espera-se um bom entendimento sobre o problema a ser tratado

Design (Idea generation)

Durante essa fase, a busca e a produção de uma solução são os principais focos. A principal atividade realizada nessa etapa é o brainstorming, onde busca-se obter múltiplas ideias e soluções. Esse tipo de atividade incentiva ideias criativas e que consequentemente poderão ser inovadoras e diferentes das atuais soluções. Outras atividades comuns nessa etapa incluem storyboarding, card sorting e diagrama de afinidade. Durante a idealização do conceito pode-se adotar diferentes posturas de design como o design gráfico, design de interação, user experience, etc. que ajudarão a moldar a forma e comportamento do produto.

Prototipação (Prototyping)

Protótipos podem ser construídos através das ideias e conhecimentos obtidos durante a etapa de design. Além disso, a construção de protótipos também pode contribuir com a criação de novas ideias. Produzir o produto de verdade é uma tarefa que frequentemente exige um grande esforço. Um protótipo pode ser construído, por exemplo, somente com papel e objetos de um ambiente de escritório e então testada. Para sistemas digitais, um outro tipo comum de protótipo é o *Mágico de Oz*. Nesse tipo de protótipo, o sistema parece estar em real funcionamento para quem usa, mas na verdade está sendo manipulado e comandado por uma pessoa sem que o usuário saiba. Por exemplo, um sistema que reconhece comandos de voz pode usar esse tipo de protótipo sem que haja de fato um software que reconheça a voz e a traduza em palavras escritas. Uma pessoa ouvirá o que o usuário disser e fará o trabalho de tradução que o sistema deveria fazer sem que o usuário perceba.

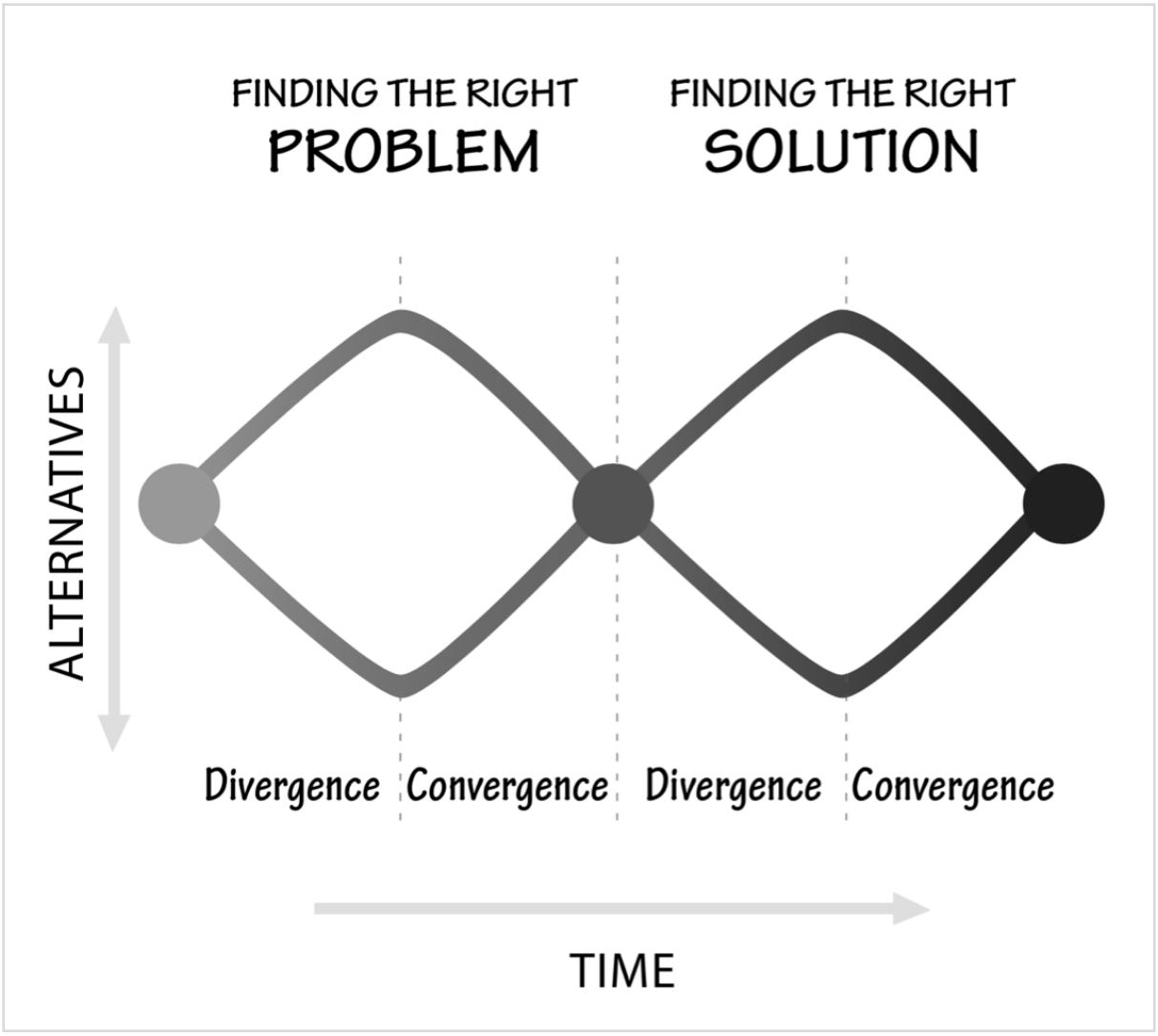
Teste (Testing)

Os protótipos criados devem ser testados e avaliados. O principal ponto dessa etapa é envolver os usuários e fazê-los usar os protótipos. A partir de métodos formais de usabilidade, enquetes, observação, etc. busca-se compreender as dificuldades e formas com que o usuário fez uso do protótipo. Entender o que deve ser mudado para melhorar o produto ou ainda mesmo, mudar completamente o projeto são informações cruciais a serem obtidas. Assim que terminada essa fase, completa-se uma iteração e volta-se à etapa de pesquisa. Isso permite um contínuo refinamento e melhoria do produto sendo desenvolvido.

O processo de prototipação e teste também pode ser utilizado quando o problema está sendo estudado na fase de pesquisa. Dessa forma pode-se ter mais certeza de que o problema que o produto busca resolver foi bem compreendido.

Discussão

Ao colocar o usuário no centro de todas as decisões de design e incluí-lo em várias etapas do processo de desenvolvimento, problemas enfrentados por reais usuários serão percebidos antes e mais facilmente. Parte das vantagens do método de UCD acontece por conta do grande foco em entender o problema a ser resolvido antes de procurar por soluções. Isso remete bastante ao *modelo double-diamond de design* introduzido pelo Conselho Britânico de Design em 2005:



(Norman, 2013)

A partir de uma ideia realiza-se pesquisas de design para expandir o espaço de problemas. Busca-se explorar novas perspectivas e obter maior entendimento sobre o problema a ser tratado. Nesse momento há uma grande divergência de ideias. Após a compreensão das questões fundamentais ao projeto, passa a se convergir para a formulação de um problema. A partir dele, uma nova fase de divergência ocorre explorando diversas soluções e caminhos inusitados para só então, convergir em uma solução.

Em Brown (2010), o autor descreve como no momento das fases de divergência passada por designers, demais equipes tendem a se assustar e se preocupar com o desenvolvimento do design. Designers parecem estar fugindo do foco do projeto e as vezes caminhando em direção oposta à solução. O pensamento convergente é essencial para decidir entre múltiplas ideias e funciona como um funil. No entanto para criar ideias inovadoras e disruptivas o pensamento divergente se mostra muito mais promissor.

Em áreas voltadas à tecnologia como a computação, engenharia, etc. há uma tendência em encarar o desenvolvimento de um projeto partindo-se direto para o espaço de solução (losango direito) e muitas vezes pela fase de convergência. No entanto um melhor entendimento sobre o problema a ser tratado é essencial para melhor atender as necessidades do usuário. É bastante fácil achar um software que tenta fazer uso da tecnologia de forma inovadora, mas resolvendo o problema errado. Imagine o caso da locomoção urbana através de metrô. Em uma cidade com uma malha metroviária muito grande é muito difícil uma pessoa decorar o nome e localização de todas as estações. Logo, o uso de um mapa passa a ser essencial. Uma pessoa pode vir a pensar nesse problema e concluir que mapas são grandes e difíceis de serem carregados. Portanto criar um aplicativo para celular que seja um mapa parece ser a solução ideal, visto a portabilidade do celular. No entanto, o real problema do usuário não é em relação à portabilidade e sim em chegar do ponto A ao B. Uma solução mais elegante poderia ser um aplicativo que dado a estação de partida e a de destino, exibe a direção que o usuário deve seguir na linha atual e as eventuais estações de baldeações que ele deve fazer. Mais que isso um melhor estudo do problema pode vir descobrir que além dos usuários desejarem ir da estação A à B, eles querem ir da forma mais rápida, barata, com menos baldeações, segura, etc.

UCD reflete bastante o *modelo double-diamond de design* em suas diversas etapas. Durante a fase de pesquisa, o espaço do problema é estudado e através de diversas atividades, busca-se melhor entender o problema por múltiplas perspectivas. Na fase de design a divergência feita pelos processos de design, como o brainstorming, ocorre claramente. Busca-se uma solução através da explorações de múltiplos caminhos antes de começar a convergência.

A iteração é uma característica crucial de UCD. É comum que requisitos cruciais surjam durante o processo de desenvolvimento de um produto, e isso é difícil de se antecipar. Através de um processo iterativo é possível constantemente refinar as ideias ou adotar novas. Eventualmente pode-se perceber que as ideias atuais não atendem as reais necessidades de usuários e uma completa troca de direção do projeto também pode ocorrer.

4. Design Centrado no Usuário e Desenvolvimento de Software

Design Centrado no Usuário e Engenharia de Requisitos

Além do próprio problema do design de um produto, desenvolver um software inclui mais complexidade ao projeto. No desenvolvimento de software, transformar ideias em códigos que se integram em um grande sistema com múltiplas classes, bancos de dados e servidores já é um desafio por si só. Por outro lado, tais ideias são resultados de um árduo processo de design que um grupo de pessoas teve ao tentar solucionar problemas enfrentados por usuários. Integrar essas duas distintas tarefas garantindo que os múltiplos times do projeto caminhem em uma mesma direção e que a comunicação entre eles seja efetiva é, portanto, uma tarefa com um considerável maior grau de dificuldade. O design centrado no usuário busca tratar esse desafio de diversas formas.

Em UCD, uma das principais tarefas do designer é compreender as diversas concepções sobre o sistema dos diversos stakeholders, time de marketing e outras equipes. A partir disso, o designer deve ser capaz de equilibrar todas essas concepções, necessidades dos usuários, restrições tecnológicas, tempo e dinheiro disponível em um bom design. Além disso ele deve ser capaz de comunicar esse conceito para os stakeholder e demais equipes. De acordo com Gulliksen et al. (2003) um dos princípios de UCD é o uso de representações simples do design, isto é, o design deve ser representado em uma forma tal que usuários e stakeholders irão facilmente entende-lo. UCD foca principalmente na área de design e trata os requisitos do usuário como parte do processo exploratório do design. Com uma abordagem mais voltada à engenharia de software, uma alternativa a UCD para integrar o design com o desenvolvimento de software é a *engenharia de requisitos*.

Engenharia de requisitos é uma disciplina que compartilha muitos conceitos, técnicas e preocupações de UCD. Suas principais atividades são dividas em: elicitação, análise/negociação, documentação e validação. De forma semelhante à fase de pesquisa de UCD, o principal objetivo durante a elicitação é entender o problema a ser resolvido e o seu escopo. Isso é feito principalmente através de discussões com stakeholders do projeto para determinar os requisitos a serem implementados. Como durante essa fase eventuais conflitos e inconsistências entre os requisitos podem surgir, uma fase onde requisitos são analisados e futuramente negociados com os stakeholders mostra-se necessária. Após os requisitos serem selecionados, documenta-se eles para que sejam usados em etapas futuras e para a comunicação entre equipes. Antes de serem implementados, os requisitos

devem ser validados de forma a verificar se eles atendem correta e completamente às expectativas dos usuários (Gonçalves, 2017).

Ambos design centrado no usuário e engenharia de requisitos ajudam a tratar o problema de integrar o design de um sistema com o seu desenvolvimento. Engenharia de requisitos é mais voltada ao lado de engenharia de software, enquanto UCD é mais voltada para o lado de design. Apesar de ambos fazerem uso de atividades similares, como entrevistas com stakeholders, prototipação, cenários de uso, etc., a abordagem feita por cada um os diferencia. De acordo com Sutcliffe (n.d.) a engenharia de requisitos tem uma abordagem de engenharia e muito mais sistemática que a abordagem de interação humano computador (que de acordo com a conotação usada pelo autor aqui nos referenciamos por UCD). Em contraste com o processo de desenvolvimento "especificação-design-implementação" mais linear da engenharia de requisitos, UCD passou a se tornar menos *método-orientada*. Gulliksen et al. (2003) argumenta que um dos princípios do design centrado no usuário é sua capacidade de ser customizado. Apesar de UCD poder ser utilizado em uma grande variedade de situações, as suas atividades, métodos e ferramentas devem ser mudados para melhor atender cada uma dessas situações.

Os métodos em UCD, ao invés de serem vistos como receitas passo a passo com um objetivo final claro, são normalmente vistos como ferramentas que auxiliam o processo de design, de forma a estimular o lado criativo e inovador do designer. Além disso, Sutcliffe (n.d.) argumenta que uma das principais vantagens de UCD sobre a engenharia de requisitos é a inclusão da experiência de usuário no seu processo, passando a considerar emoções e sentimentos como requisitos importantes para o desenvolvimento de sistemas.

Design Centrado no Usuário e Métodos Ágeis

Nessa seção estudar-se-á como o design centrado no usuário auxilia o desenvolvimento de software, mais especificamente para casos de desenvolvimento utilizando metodologia ágil. De acordo com Hussain, Slany e Holzinger (2009), apesar de não haver muitos estudos sobre a integração dessas duas metodologias, UCD vem sendo adotado por empresas que utilizam metodologia ágil. Apesar de métodos ágeis e UCD se assemelharem com respeito ao grande foco no usuário e no consumidor, eles também possuem certas diferenças que tornam a sua integração não tão trivial. UCD tende a investir grande parte dos seus recursos em pesquisa e análise antes do desenvolvimento, podendo levar até meses para concluir completamente o design. Isso se contrapõe ao sistema de entregas rápidas de software funcional, adotada por métodos ágeis. Oura diferença crucial é no objetivo final de cada um. O maior foco da metodologia ágil é

entregar software funcional. Um sistema usável é excelente, mas é diferente de software com boa usabilidade, objetivo de UCD. Ainda assim, segundo Hussain et al. (2009), empresas de desenvolvimento de software ao redor do mundo tem obtido resultados positivos da integração de UCD com métodos ágeis.

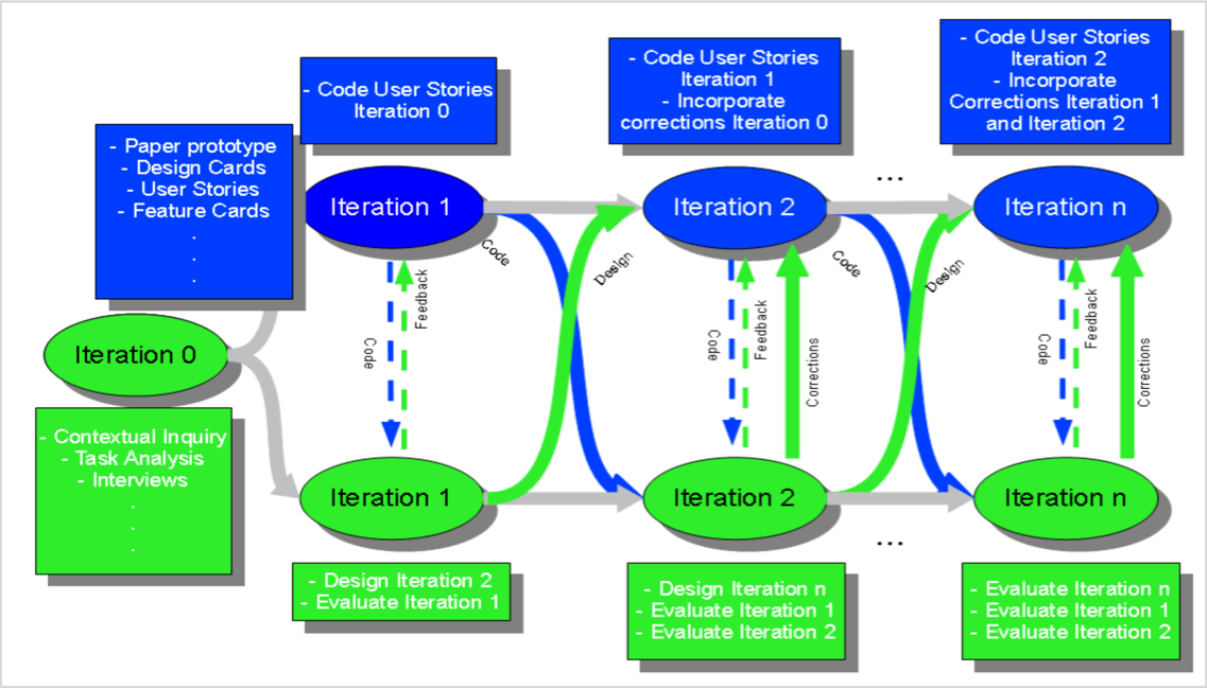
Gulliksen et al. (2003) relata que métodos ágeis inclusive solucionam certos problemas encontrados por ele durante aplicação de UCD em um projeto real. Designers em meio ao processo de desenvolvimento podem perder a visão da entrega do produto final, mas as entregas rápidas de software funcional, frequentemente enfatizadas durante processos ágeis, podem tratar essa complicação. Por outro lado times que perdem a noção da *big picture* (perspectiva geral do projeto) têm sido notados como um problema da metodologia ágil podendo também afetar equipes que utilizam UCD. O mal entendimento da documentação de design pode vir a ser solucionado através do princípio de alta comunicação humana abordada por métodos ágeis (pessoas são mais importantes que processos e ferramentas). Modelos, protótipos e outras técnicas passam a ser utilizados mais como forma de comunicação e menos como documentação.

Sintonizar as iterações de equipes de design com as *sprints* de métodos ágeis é um dos maiores desafios da união dos dois métodos. Em uma pesquisa realizada por Da Silva, Martin, Maurer e Silveira (2011) 58 artigos sobre a união de UCD com métodos ágeis foram analisados e artefatos, práticas e necessidade comuns entre eles foram identificadas. Dentre os 15 itens levantados 4 deles se referem ao tempo em que as iterações do design acontecem em relação às sprints, sendo o item mais comentado dentre os 15, o *Little Design Up Front* (LDUF), comentado por mais de 30 artigos. A prática de LDUF implica em parte do trabalho do design ser feito antes do início do desenvolvimento do sistema, isto é, na sprint 0. Não parece haver um consenso sobre quais das atividades de UCD devem ser realizadas a priori, mas contanto que não seja um trabalho extenso de design, sua prática é altamente recomendada. Ao design extenso durante a sprint 0 se dá o termo BDUF (*Big Design Up Front*) e contrário ao LDUF, é altamente não recomendado, pois vai contra os princípios ágeis. De forma semelhante ao LDUF, mais de 10 autores defendem o método *One Sprint Ahead*, ou seja além da primeira iteração do design, as demais também acontecem uma sprint a frente da sprint do desenvolvimento.

Close Collaboration, isto é, a colaboração próxima entre equipes de design e desenvolvimento aparece como segundo mais citado nessa pesquisa. Essa colaboração implica em não somente melhor comunicação entre equipes, como também em desenvolvedores entendendo melhor o que designers querem dizer. Pelo ponto de vista de UCD, para que o projeto tenha sucesso é necessário que todas as pessoas envolvidas

no desenvolvimento do projeto estejam comprometidas com a importância da usabilidade no sistema. Gulliksen et al. (2003) descreve que um dos problemas na prática de UCD é que estabelecer essa atitude centrada no usuário em toda a equipe é trabalhoso e eventualmente designers de usabilidade passaram a ser ignorados. Espera-se que através de uma melhor colaboração entre equipes esse tipo de problema seja sanado.

Outros pontos ressaltados incluem o uso de técnicas como personas, cenários e história de usuários e são melhores contemplados nos artigos analisados por Da Silva et al. (2011). Questões sobre quando realizar a prototipação, testes de usabilidades e inspeções de usabilidade foram levantadas e o seguinte framework foi proposto por Da Silva et al. (2011):



(Da Silva et al., 2011)

A cor azul representa o time de desenvolvedores e o verde o de designers. Na iteração 0, cada equipe realiza as atividades descritas na imagem como forma de preparo para a iteração 1. Durante a iteração 1, a equipe de design além de realizar o mesmo tipo de atividade de design (preparo para a próxima iteração), também realiza avaliações de usabilidade no sistema sendo criado pela equipe de desenvolvimento. Um feedback é dado para os desenvolvedores na forma de *oral storytelling* para a mesma iteração e protótipos, cartas de design e histórias de usuário para a iteração seguinte. A partir da iteração 2, além de realizar as mesmas atividades a equipe de design realiza teste de usuário com o sistema desenvolvido na iteração anterior. A equipe de desenvolvedores corrige os erros apontados pela equipe de design na iteração anterior.

Discussão

Ao se falar em desenvolvimento de software costuma-se pensar somente em tarefas que envolvem a parte técnica como algoritmos, organização de classes e outras estruturas, bancos de dados, servidores, etc. No entanto não seria o design, isto é, projetar o conceito de um software, uma atividade de desenvolvimento de software também?

Imagine um mundo onde não há programação e um programa é criado instantaneamente da forma desejada. Nesse mundo, não há a necessidade de se preocupar sobre que classes criar, como integrar bancos de dados, servidores, etc. tudo acontece magicamente a partir de uma ideia. No entanto, nesse mundo, o desafio de desenvolver um software ainda não é nulo. O pensamento mais comum de um desenvolvedor de software nesse momento talvez seja de que só resta, então, pensar em como será a aparência desse sistema. De fato, ainda resta pensar na aparência, mas se só isso for feito, produzir-se-á apenas uma imagem estática. Deve-se levar em consideração o que acontecerá quando um usuário interagir com um software, isto é, de que forma ele irá mudar quando alguém clicar em um botão, etc. Além disso pode-se considerar também os sentimentos e emoções do usuário ao usar esse sistema (ele ficará feliz quando usar?, se sentirá seguro?, etc.). Parece, também, razoável supor que esse sistema irá resolver algum problema de um usuário e que irá fazer isso de forma efetiva. Em outras palavras, nesse mundo ideal, ainda faltaria considerar o design da interface, design de interação, experiência do usuário e a usabilidade, de forma a resolver o problema do usuário (que pode ser conhecido ou não). UCD nada mais é que um processo que oferece um processo para se realizar todas essas tarefas. Engenharia de requisitos apresenta-se como uma solução alternativa para o mesmo processo, porém possui suas distinções, como citado anteriormente.

Na maioria dos projetos, para cada uma das áreas citadas (UI, IxD, UX, usabilidade, UCD) não costuma-se haver um especialista, ou uma equipe, que se dedica exclusivamente. Em projetos de software é comum que ele inteiro seja desenvolvido somente por programadores, que realizarão tanto a parte de design quanto a de programação. Eventualmente pode ser contratado um designer gráfico, que irá ajudar a fazer uma *"interface bonita"*, e as demais tarefas continuarão sendo do desenvolvedor, que costuma não ter o treinamento adequado. Apesar da necessidade de um profissional capaz de realizar o design de sistemas interativos ter sido notada desde a década de 1990, poucas universidades ao redor do mundo oferecem graduações ou pós-graduações em design de interação. Profissionais de outras áreas, como no exemplo anterior, são obrigados a realizar atividades para as quais não foram totalmente preparados. Cabe a tais

profissionais, como o designer e o desenvolvedor, obter conhecimento sobre áreas como o design de interação, interação humano computador, user experience, usabilidade e design centrado no usuário e não tomá-las como triviais ao serem designados para realizar atividades relacionadas.

Voltando ao exemplo anterior de mundo ideal, onde softwares são construídos como mágica, se tal suposição for retirada, isto é, considerando que trata-se do mundo real, onde transformar ideias em software funcional é um trabalho custoso, surgir-se-á o problema de implementação. Software leva tempo para ser programado, exige muitas pessoas e recursos como servidores, data centers, etc. e é um problema muito bem tratado pela engenharia de software, com métodos como *métodos ágeis*. Portanto, modelos como o apresentado na seção anterior são essenciais para qualquer projeto de software que queira adotar o design centrado no usuário.

Conclusão

A proposta desse trabalho era estudar a fundo o design centrado no usuário e como ele está relacionado com a computação. Acredita-se que ao final desse trabalho, todos os objetivos estipulados foram alcançados. Através do estudo da história de interação humano computador, foi possível identificar como surgiu UCD. Pelo fato de UCD não ser utilizado exclusivamente para o desenvolvimento de software e enfatizar diversas ideologias não comuns ao campo da computação, esperava-se que suas raízes fossem em campos de estudo como o do design. No entanto foi uma surpresa descobrir que seu nascimento ocorreu devido ao uso de computadores e dentro de IHC, uma disciplina, em partes, da computação.

Observou-se que o estudo da usabilidade de sistemas já acontece há muitas décadas e acompanha a computação desde os anos 70. A criação de computadores pessoais e a adoção de computadores no ambiente de trabalho impulsionaram os estudos de IHC, que passou a crescer na comunidade científica. Uma enorme mudança no mundo da computação ocorre com a invenção da internet. O foco no uso do computador passa a ser comunicação e o número de usuários sem conhecimento técnico começa a a crescer rapidamente. Consequentemente, o mercado de computadores (de forma geral) passa a crescer e usabilidade se torna uma das maiores preocupações. Começa a se notar que o aspecto visual do software também é relevante trazendo designers para o mundo de desenvolvimento de software. Designers ajudam a mudar as perspectivas de IHC e user experience passa a ser uma preocupação de IHC.

Esse caminho seguido pela computação não é incomum em relação a outras tecnologias. No princípio, tecnologias são usadas por um grupo muito pequeno de pessoas e muitas vezes essas se adaptam às dificuldades encontradas ao usar o produto. Com o aumento de seu uso e a sua popularização, mais pessoas passam a usufruir dessa tecnologia e preocupações a respeito de eficiência e usabilidade passam a ser discutidas. Vivemos uma fase na qual softwares começam a ter maior concorrência e isso os estimula a se diferenciarem e terem auto expressão. Atualmente parece não haver um consenso sobre qual o profissional responsável por realizar o design de um software (não somente da parte visual) e qual o escopo do seu papel. No entanto há uma preocupação em incluir algum tipo de profissional que faça esse design se preocupando com as interações com o produto e em como melhor atender as necessidades do usuário. A prática de UCD busca tratar esse problema através de um processo iterativo de design.

Em prol de identificar as necessidades e objetivos dos usuários, UCD faz um intenso

estudo sobre o problema a ser tratado antes de explorar soluções. O designer busca equilibrar objetivos dos diversos stakeholders, tempo e orçamento com as necessidades de usuários para criar o produto. Durante a fase de design, busca-se explorar múltiplas ideias antes de escolher uma, de modo a incentivar a criatividade e inovação. O usuário é incluído nas diversas etapas para um entendimento sobre a real eficiência do produto.

No desenvolvimento de software, uma abordagem mais comum, bastante ligada à engenharia de software, é a engenharia de requisitos. Apesar de bastante semelhante à UCD, engenharia de requisitos tem métodos muito mais rígidos e sistemáticos que podem vir a ser prejudiciais ao processo criativo e com olhar mais subjetivo do design. Além disso uma das maiores vantagens de UCD é a inclusão de UX durante o processo.

Múltiplos autores comentam que integração de métodos ágeis com UCD ainda é muito pouco estudada. Durante essa integração, segundo o modelo proposto por Da Silva et al. (2011), a equipe de design deve trabalhar uma sprint a frente da equipe de desenvolvimento. Grande atenção deve ser dada à comunicação entre equipes, portanto protótipos, cenários, história de usuários e outras técnicas servem mais a esse propósito do que para documentação.

Graças aos estudos feitos nesse trabalho, obteve-se uma visão geral sobre o que é UCD e o seu escopo, principalmente dentro da computação. Acredita-se que o entendimento sobre o contexto em que essa área se encontra e sua relação com a computação é importante para compreender o tipo de pesquisa sendo realizada e o rumo que ela pode vir a seguir. Apesar de UCD ter mais de 30 anos de existência, e de sua grande importância para desenvolver softwares que atendam as necessidades de seus usuários, ela ainda é pouco disseminada pela comunidade de desenvolvedores de software. Espera-se que nos anos que estejam por vir, isso continue a mudar, como vem acontecendo nos últimos anos, e que a comunidade da computação passe a valorizar não somente a parte técnica do desenvolvimento de software, mas também a parte que projeta a ideia de um software pensando em quem irá utilizá-lo.

Referências

1. Brown, T. (2009). *Change by Design*. New York: HARPER COLLINS.
2. Buxton, B. (2007). *Sketching user experiences: getting the design right and the right design*. Morgan Kaufmann.
3. Carroll, J. M. (n.d.), *Human Computer Interaction - brief intro*. Retrieved from https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/human-computer-interaction-brief-intro
4. Carroll, J. M. (2001). The evolution of human-computer interaction. *Annual Review of Psychology*, *48*, 501-522.
5. Cooper, A., Reimann, R., & Cronin, D. (2007). *About face 3: The essentials of interaction design*. Indianapolis, IN: Wiley.
6. Da Silva, T. S., Martin, A., Maurer, F., & Silveira, M. (2011, August). User-centered design and agile methods: a systematic review. In *Agile Conference (AGILE), 2011* (pp. 77-86). IEEE.
7. Kapor, M. (1996). A software design manifesto. In *Bringing design to software* (pp. 1-6). ACM.
8. Gonçalves, J. A. M. (2017). *Requirements engineering in software startups: a qualitative investigation* (Doctoral dissertation, Universidade de São Paulo).
9. Grudin, J. (2012). A Moving Target: The evolution of Human-computer Interaction. In J. Jacko (Ed.), Human-computer interaction handbook: Fundamentals, evolving technologies, and emerging applications. (3rd edition). Taylor & Francis
10. Gulliksen, J., Göransson, B., Boivie, I., Blomkvist, S., Persson, J., & Cajander, Å. (2003). Key principles for user-centred systems design. *Behaviour and Information Technology*, *22*(6), 397-409.
11. Hussain, Z., Slany, W., & Holzinger, A. (2009, November). Current state of agile user-centered design: A survey. In *Symposium of the Austrian HCI and Usability Engineering Group* (pp. 416-427). Springer, Berlin, Heidelberg.
12. Iivari, J., & Iivari, N. (2006). Varieties of user-centeredness. In *System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on* (Vol. 8, pp. 176a-176a). IEEE.
13. Maguire, M. (2001). Methods to support human-centred design. *International journal of human-computer studies*, *55*(4), 587-634.
14. Myers, B. A. (1998). A brief history of human-computer interaction technology. *interactions*, *5*(2), 44-54.
15. Norman, D. A. (2013). *The design of everyday things: Revised and expanded edition*. New York: Basic Books.
16. Norman, D. A., and Draper S. W. (1986). *User Centered System Design: New Perspectives on Human-Computer Interaction*. New Jersey: Lawrence Erlbaum Associates.

17. Preece, J., Rogers, Y., & Sharp, H. (2015). *Interaction design: beyond human-computer interaction*. John Wiley & Sons.
18. Ritter, F. E., Baxter, G. D., & Churchill, E. F. (2014). *Foundations for designing user-centered systems: What system designers need to know about people*. Springer.
19. Sutcliffe, A. G. (n.d.), *Requirements Engineering*. Retrieved from https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/requirements-engineering)
20. Venturi, G., Troost, J., & Jokela, T. (2006). People, organizations, and processes: An inquiry into the adoption of user-centered design in industry. *International Journal of Human-Computer Interaction*, *21*(2), 219-238.
21. Williams, A. (2009, October). User-centered design, activity-centered design, and goal-directed design: a review of three methods for designing web applications. In *Proceedings of the 27th ACM international conference on Design of communication* (pp. 1-8). ACM.
22. Winograd, T. (1996). From Computing Machinary To Interaction Design. In *Beyound Calculation: the next fifty years of computing* (pp. 149-162) Amsterdam: Springer–Verlag/
23. Zimmerman, J., Forlizzi, J., & Evenson, S. (2007). Research through design as a method for interaction design research in HCI. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 493-502). ACM.

Parte subjetiva

Acredito que devido ao contato com diversos programas com péssima usabilidade eu tenha me atraído para assuntos relacionados a design, usabilidade, etc. Sempre tive interesse em cursar a matéria MAC0446 Princípios de Interação Humano-Computador, mas parece que desde 2013, em todos os semestres que eu tentei não estava sendo oferecida (parte da minha graduação passei em intercâmbio, então não estive em todos os semestres no IME). Não sei qual a frequência de oferecimento nem a procura, mas acredito que é uma matéria importante e deveria ser mais oferecida e divulgada.

Foram raríssimas as vezes em que eu ouvi falar de design centrado no usuário durante minha graduação. Em MAC0426 Sistemas de Bancos de Dados e MAC0350 Introdução ao Desenvolvimento de Sistemas de Software aprende-se muito pouco sobre analise de requisitos, porém nenhuma atenção é dada ao pensamento centrado no usuário e suas necessidades. Uma das preocupações de alguém que faz computação é em fazer software de qualidade e, em minha concepção, qualidade inclui uma boa usabilidade e experiência para o usuário, não somente qualidade de código. No entanto parece que durante a graduação não se dá o devido valor a esse outro lado do desenvolvimento de software. Existe uma grande ênfase no aspecto técnico de como transformar uma ideia em software, mas pouco sobre como desenvolver plenamente essa ideia.

MAC0472 - Laboratório de Métodos Ágeis é uma disciplina onde se desenvolve um sistema real com um cliente real. É uma disciplina que ensina bastante sobre como de fato funciona o desenvolvimento de software numa situação real. Ainda assim, durante essa disciplina, design continua sendo a atividade de *"desenhar uma interface bonita"*. É claro que o foco da disciplina não é somente o processo do design, mas existe uma parcela considerável de tempo discutindo com o cliente o projeto a ser desenvolvido. Nesse momento um processo para se desenvolver a ideia do software parece mais do que necessário e o design centrado no usuário, mesmo que de forma simplificado, parece prover formas de ajudar. Ainda, acredito que uma matéria com a mesma estrutura de MAC0472, porém com enfoque na prática do design centrado no usuário poderia ser muito interessante.

Durante esse trabalho foi citado o *modelo double-diamond de design*, que discute o espaço do problema e o espaço de solução. Quando você vai sair de que casa e está preocupado em esquecer algo (principalmente antes de viagens), você soluciona isso

pensando no espaço do problema ou de solução? Eu passei a notar que antigamente eu buscava em meu quarto objetos que eu eventualmente pudesse estar esquecendo. Já agora, eu penso no que eu farei durante o dia, ou dias, e me imagino nessa situação (como se eu estivesse aplicando a técnica de cenários de uso em mim mesmo). Então, eu penso: nesse cenário, que objetos eu estarei usando, e quais são extremamente necessários para realizar tal tarefa? Eventualmente eu acabo me lembrando de algum ou outro objeto e na pior das hipóteses, eu esqueço algum objeto que não será extremamente necessário para o que eu estarei fazendo. Acredito que essa forma de resolver o problema, analisando o espaço do problema tem se mostrado efetiva.
