Universidade Estadual de Londrina

ANDRÉ FELIPE BERGAMIM

FRAMEWORK DE CONVERGÊNCIA DE DESENVOLVIMENTO ÁGIL E DESIGN THIKING:

ESTUDO APLICADO

Londrina
2018

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UEL

Bergamim, André Felipe.

Framework de convergência de desenvolvimento ágil e design thinking: estudo aplicado / André Felipe Bergamim. - Londrina, 2018. 101f.: il.

Orientador: Prof. Dr. Rodolfo Miranda de Barros.
Dissertação (Mestrado em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Programa de Pós-Graduação em Ciência da Computação, 2018.
Inclui bibliografia.

1. Framework - Tese. 2. Desenvolvimento Ágil - Tese. 3. Design Thinking - Tese. 4. Gerenciamento de Projeto - Tese. I. Barros, Rodolfo Miranda. II. Universidade Estadual de Londrina. Centro de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação. III. Título.

ANDRÉ FELIPE BERGAMIM

FRAMEWORK DE CONVERGÊNCIA DE DESENVOLVIMENTO ÁGIL E DESIGN THIKING:

ESTUDO APLICADO

Dissertação apresentada ao Programa de Mestrado em Ciência da Computação da Universidade Estadual de Londrina, para a obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Rodolfo Miranda de Barros.

Londrina
2018

ANDRÉ FELIPE BERGAMIM

FRAMEWORK DE CONVERGÊNCIA DE DESENVOLVIMENTO ÁGIL E DESIGN THIKING:

ESTUDO APLICADO

Dissertação apresentada ao Programa de Mestrado em Ciência da Computação da Universidade Estadual de Londrina, para a obtenção do título de Mestre em Ciência da Computação.

BANCA EXAMINADORA

_____

Orientador: Prof. Dr. Rodolfo Miranda de Barros
Universidade Estadual de Londrina - UEL

_____

Prof. Dr. Alan Salvany Felinto
Universidade Estadual de Londrina - UEL

_____

Prof. Dr. Lourival Aparecido Góis
Universidade Tecnológica Federal do Paraná - UTFPR

Londrina, 26 de abril de 2018.

AGRADECIMENTOS

Primeiramente, agradeço a quem me motivou e sempre me apoiou de diversas formas durante minha vida e agora neste momento - minha mãe Luiza Bergamim, que diariamente me acompanhou e que se aqui estou, na reta final de minha pós-graduação, foi devido a este apoio.

Na sequência agradeço aos meus amigos de turma, companheiros e cúmplices nesta difícil jornada, que tornaram o peso do dia-a-dia mais leve e divertido.

Agradeço ao meu orientador, professor Rodolfo de Barros e a professora Vanessa Tavares, pelas contribuições, reflexões e principalmente pela confiança e apoio pelos muitos obstáculos que tive ao longo do mestrado.

Incluo também os professores, componentes de banca, que se disponibilizaram a ler este trabalho e contribuir com sua composição.

Finalmente, agradeço a todos os autores de artigos e livros consultados, que compõe minha bibliografia e o corpo deste trabalho, sem eles, com certeza, este trabalho não existiria.

“Quoth the raven:
Nevermore!"
Edgar Allan Poe

BERGAMIM, A. F. **Framework de convergência de desenvolvimento ágil e design thinking: estudo aplicado**. 101f. Dissertação de Mestrado (Mestrado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2018.

RESUMO

O desenvolvimento de software é uma atividade de intenso exercício de conhecimento e um esforço colaborativo de diferentes áreas, portanto uma empresa deve proporcionar processos, métodos e ferramentas como parte do ambiente de produção. Entretanto, a implementação de tais estruturas para gerenciar o desenvolvimento de software constitui em um grande desafio, pois além de ser uma tarefa árdua de convergência de saberes, este exercício pode representar obstáculos no desenvolvimento de um projeto. Visto que a criação de um software é uma atividade intensa de inovação e criatividade, o design passou cada vez mais a integrar as equipes de desenvolvimento, porém nota-se uma tendência em abordar esta disciplina somente em nível operacional, como uma ferramenta para definir metas e agregar valores estéticos ao produto final. Partindo disto, esta pesquisa busca delimitar como as teorias e práticas de design e tecnologia da informação estão incorporadas no gerenciamento de projeto de software e propor um framework de trabalho de convergência destas duas áreas. A construção do framework teve o aporte da literatura acerca do tema, levantamento de dados dos relatórios de projetos de software e questionários com especialistas. Os resultados foram então utilizados para analisar o framework atual do Laboratório Gaia e propor um novo modelo de processo com o design inserido em seu programa. Por fim, o trabalho foi avaliado por especialistas e os resultados iniciais indicam o potencial do modelo para melhoria de resultados e alinhamento do processo de desenvolvimento no qual design e TI trabalhem de forma colaborativa.

Palavras-chave: Framework. Desenvolvimento Ágil. Design Thinking. Desenvolvimento Software. Design. Gerenciamento de Projeto.

BERGAMIM, A. F. **Agile development and design thinking convergence framework: applied study**. 101p. Master's Thesis (Master in Science in Computer Science) – State University of Londrina, Londrina, 2018.

ABSTRACT

Software development is an activity of intense knowledge exercise and a collaborative effort of different areas, therefore a company must provide methods and tools as part of the creation environment. However, the implementation of such controls and structures is a major challenge, for it may represent obstacles in the development of a project. Since the creation of a software is an intense activity of innovation and creativity, design has increasingly integrated development teams, but there is a tendency to approach it only at the operational level, as a tool to set goals and add aesthetic values to the final product. Based on this, this research seeks to delimit how the theories and practices of design and information technology are incorporated in the management of software projects and propose a convergence framework of these two areas. The construction of the framework was supported by literature on the subject, data collection of software project's reports and questionnaires with specialists. The results were then used to analyze the current framework of Gaia Laboratory and propose a new process model with the design embedded in its structure. Finally, the work was evaluated by experts and the initial results indicate the potential of the model to improve results and alignment of the development process in which design and IT work collaboratively.

Keywords: Framework. Agile Development. Design Thinking. Software Development. Design. Project Management.

LISTA DE ILUSTRAÇÕES

Figura 1: Modelo tradicional 22
Figura 2: Modelo iterativo e incremental 23
Figura 3: Método cascata 24
Figura 4: Modelo espiral [67] 25
Figura 5: Scrum task board [57] 30
Figura 6: eXtreme programming [32] 30
Figura 7: IDEO design thinking [47] 35
Figura 8: Stanford design thinking [34] 35
Figura 9: Zona de fluxo [64] 49
Figura 10: Google material [37] 53
Figura 11: GAIA PDS [21] 54
Figura 12: Processo app câmbio 59
Figura 13: Cronograma app ensino de inglês 63
Figura 14: Fases de maior impacto 65
Figura 15: GAIA agile 68
Figura 16: GAIA iterações 69
Figura 17: GAIA agile framework 71
Figura 18: Convergência TI e DT 72
Figura 19: Função do design 75
Figura 20: GAIA framework detalhado 76
Figura 21: Mapa conceitual 80
Figura 22: Personas 81
Figura 23: Scrum board 82
Figura 24: Jornada do usuário 82
Figura 25: Fluxo de telas 83
Figura 26: Storyboard 83
Figura 27: Análise heurística guiada 84
Figura 28: Roda de ferramentas 85

LISTA DE TABELAS

Tabela 1: Qualidade de software: framework tradicional x ágil [24] 28
Tabela 2: Design vs design thinking [52] 34
Tabela 3: Heurísticas de Nielsen [44] 45
Tabela 4: Análise inicial 55
Tabela 5: Análise e planejamento 56
Tabela 6: Execução e implementação 56
Tabela 7: Validação e testes 57
Tabela 8: Entrega 57
Tabela 9: Parâmetros definidos 67
Tabela 10: Fases GAIA agile 69
Tabela 11: Ferramentas de design thinking 73
Tabela 12: Briefing 77
Tabela 13:Resultados da aplicação do questionário qualitativo 86

LISTA DE ABREVIATURAS E SIGLAS

DT	Design Thinking
IHC	Interação Humano-Computador
UI	User Interface
UX	User Experience
TI	Tecnologia da Informação

SUMÁRIO

1 INTRODUÇÃO 14
1.1 Justificativa 16

2 METODOLOGIA DE PESQUISA 18

3 FUNDAMENTAÇÃO TEÓRICA 19
3.1 Engenharia de Software 19
3.1.1 Controle de Qualidade 21
3.1.2 Iterações 22
3.1.2.1 Ciclo de vida: cascata 23
3.1.2.2 Ciclo de vida: espiral 25
3.1.2.3 Ciclo de vida: incremental e evolutivo 26
3.1.3 Desenvolvimento Ágil - Agile 27
3.1.3.1 Scrum 29
3.1.3.2 Extreme programming 30
3.1.4 Desafios em Projetos de Desenvolvimento de Software 31
3.2 Gestão de Design 32
3.2.1 Design Thinking 33
3.2.2 Ferramentas de Design Thinking 36
3.2.2.1 Imersão 36
3.2.2.2 Imersão em profundidade 37
3.2.2.3 Análise e síntese 38
3.2.2.4 Ideação 39
3.2.2.5 Prototipação 40
3.2.2.6 Implementação 41
3.2.3 Inovação Incremental 41
3.3 Interação Humano-Computador e Usabilidade 42
3.4 Design Thinking e Desenvolvimento Ágil 45
3.5 Gamificação como Análise da Motivação 47
3.5.1 Motivação 48

4 PROCESSOS METODOLÓGICOS 51
4.1 Análise de Similar: Google Material 51
4.2 Análise do Framework GAIA 53
4.2.1 Análise de Projeto: App Câmbio Monetário 57

4.2.2 Análise De Projeto: App Para Ensino De Inglês A Crianças 61
4.3 Questionários 64

5 RESULTADOS 67
5.1 GAIA Agile Framework 67
5.2 Convergindo Design Thinking e Gaia Agile Framework 72
5.2.1 Detalhamentos das Ferramentas 77
5.2.2 Roda de Ferramentas 84
5.3 Questionário com Especialistas 85

6 CONCLUSÃO 87

REFERÊNCIAS 89

APÊNDICE A – QUESTIONÁRIO 96

APÊNDICE B – AVALIAÇÃO FINAL 99

TRABALHOS PUBLICADOS PELO AUTOR 101

1 INTRODUÇÃO

No atual cenário da produção de conhecimento, como destaca Manfredo [36] é cada vez mais presente a demanda pela integração de diferentes áreas do saber, seja como uma resposta à hiperespecialização que marca o cenário acadêmico e científico, ou como uma exigência para a formação de profissionais mais compatíveis com as exigências do mercado de trabalho.

O desenvolvimento de software é uma atividade de intenso exercício de conhecimento. Wohlin, Šmite e Moe [68] sob esta visão afirmam que grande parte do desenvolvimento de software é um esforço colaborativo de diferentes expertises, portanto uma empresa deve proporcionar processos, métodos, técnicas e ferramentas como parte do ambiente de produção. Além disso, Šmite et al. [60] atentam para o fato de ser impossível um indivíduo possuir todo conhecimento necessário para trabalhar em projetos de tal amplitude e equipes de software precisam contar com diferentes redes de conhecimento dentro e fora da empresa.

Segundo McCaffery e Goleman [38] a implementação de controles e estruturas para gerenciar o desenvolvimento de software constitui em um grande desafio, pois além de ser uma tarefa árdua de integração de saberes, este exercício pode representar obstáculos no desenvolvimento de um projeto, afetando o seu escopo (interno e externo) de forma que como Shmueli e Ronen [58] afirmam, o risco envolvido em um projeto é função proporcional ao tamanho do processo como um todo.

Diversos autores [24][54] apontam para excelência dos resultados na redução de tempo de produção/custo e no aumento da qualidade quando frameworks de trabalho são estabelecidos. Entretanto, como evidenciado em resultados da revisão sistemática realizada por Khosrowshahi [29] estes frameworks abordam somente a integração contínua, e assim como outras pesquisas na área, possuem caráter excessivamente técnico/organizacional, não levando em consideração os processos criativos e a função de outros profissionais, como o designer.

Quando se tratando de equipes de software o que ocorre frequentemente é a confusão de terminologias utilizadas para especificar a função de determinados profissionais dentro de um projeto. O que comumente se percebe é a fusão de duas expertises na figura do desenvolvedor de software, que de acordo com alguns autores engloba em uma pessoa só as habilidades de criação de software (programação) e

também de design de interfaces. Porém a realidade como sempre se mostra mais diversa e devido à intensa exigência de um mercado cada vez mais voltado à satisfação do usuário equipes de software contam com a participação de profissionais especializados somente em uma das áreas, ou seja, contam com programadores e designers.

Seguindo este pensamento e contrapondo a ideia prevalente da engenharia de software [39] de que desenvolvimento de software significa construir sistemas cujo foco está em suas estruturas internas focado em um processo lógico e sistemático, Fernandes [17] aponta para o fato de que um software é um artefato humano, uma entidade descritiva, cognitiva e histórica, concebida através de esforços coletivos e, portanto, é também um processo altamente subjetivo e criativo.

Visto que a criação de um software é uma atividade intensa de inovação e criatividade, o design passou cada vez mais a integrar as equipes de desenvolvimento, porém nota-se uma tendência em abordar esta disciplina somente em nível operacional, como uma ferramenta para definir metas e agregar valores estéticos ao produto final, como por exemplo, as metodologias de trabalho voltadas ao usuário, as chamadas User-experience oriented, User-centered process e User-friendly approach, ou seja, o design não possui um campo de atuação bem definido e tem o seu grau de decisão ofuscado pela falta de abordagem da sua atuação em níveis táticos e estratégicos.

Portanto, estabelecer objetivos iniciais e delimitar esforços requer a compreensão dos conceitos chave que envolve um sistema a ser desenvolvido como meio de minimizar esforços [25], especialmente porque diante da miríade de nomenclaturas e especializações dentro da área de design e desenvolvimento de software as funções e o papel de cada profissional tornam-se confusa uma que vez que não é claro onde as expertises estão localizadas.

Designer de Interface, Designer Gráfico, Desenvolvedor de Software, Programador, Web Designer, User-Experience (ux) Designer, User Interface Designer são apenas algumas das titulações dos profissionais envolvidos em criação de software e não há acordo unânime qual é a função de cada um. Portanto, esta pesquisa busca delimitar como as teorias e práticas de design e tecnologia da informação (ti) estão incorporadas no gerenciamento de projeto de software, independente de qual profissional as estão executando, e propor um framework de trabalho de convergência dessas áreas.

Partindo desta premissa o trabalho está inserido no âmbito da engenharia de software, gerenciamento de projetos, gestão do conhecimento e gestão do design.

Consiste em aproximar e integrar o designer de interface junto à equipe de software (ti), por meio de instrumentos, técnicas e metodologias das duas áreas, pretende-se com este trabalho minimizar os esforços, riscos, erros e perdas orçamentárias que ocorrem no desenvolvimento e consequentemente aumentar a produtividade. Portanto no escopo deste trabalho tem-se a seguinte problemática: Como desenvolver um framework de convergência de desenvolvimento Agile e Design Thinking e aplicá-lo em um laboratório de TI?

1.1 Justificativa

O fator desencadeador deste trabalho foi a necessidade vista pelos profissionais de TI e Design de Interface, em projetos realizados pelo laboratório GAIA da universidade estadual de londrina, de uma metodologia de trabalho que organize a atuação dos dois profissionais em projetos de desenvolvimento de software em prol da qualidade, tanto em termos de programação quanto de design, focados na experiência do usuário.

Além disso, Acuña et al. [1] demonstra que a maioria das falhas em desenvolver um software é causa direta de fatores humanos. Dentre estes há uma gama enorme de problemas que podem ser estudados, como motivação, formação profissional, falha no modelo de negócios que exclui o fator humano, etc. No caso do presente trabalho o problema está justamente no relacionamento de duas áreas que, como aponta Norman e Jerrard [46] ambientes organizacionais que fazem uso de profissionais do design mas não possuem gerenciamento voltado para tal área, impõe a tais profissionais um abismo entre eles e o resto da organização pois o design fica sem meios de se relacionar com a estratégia corporacional e comunicar-se efetivamente com outras disciplinas.

Devido ao caráter pluridisciplinar da pesquisa, uma vez que envolve o estudo de duas áreas de forma não linear com objetivo de construir um conhecimento sólido para a solução de um problema imediato, a contribuição do presente trabalho para as duas áreas que aborda está justamente na redução de uma lacuna que impede a comunicação de profissionais em busca de resultados satisfatórios para o mesmo problema. Como afirma Dell'era e Verganti [12] se constata que quanto mais uma empresa de posiciona no mercado como geradora de inovação, maior o papel do design e seu grau de importância em decisões relacionadas ao mesmo

Aqui, torna-se relevante comentar a pertinência de tal pesquisa estar sendo realizada por um profissional da área de design dentro do programa de mestrado de ciência da computação. Desta forma, o problema pode ser abordado de dentro para

fora, por meio da aproximação com estudantes, profissionais e as disciplinas cursadas no programa. Além disso a colaboração que ocorre entre orientador e orientando de áreas diferentes favorece, no que diz respeito a elaboração de dois saberes, para a resolução de tal problema.

Por fim, espera-se que este projeto traga relevante contribuição à comunidade de designers e profissionais de TI, mediante o desenvolvimento de uma ferramenta que desobstrua esse canal de comunicação, propiciando excelência nos resultados obtidos, assim como despertar o interesse da comunidade acadêmica acerca do problema.

2 METODOLOGIA DE PESQUISA

Primeiramente foi realizado investigação teórica acerca de temas relacionados, por meio de pesquisa bibliográfica das metodologias mais atuais utilizadas dentro das áreas de gestão do conhecimento, design de interface e tic. A discussão teórica teve como finalidade gerar instrumentos de análise e de trabalho que foram aplicados nos estudos de casos, na análise de dados e na criação da proposta de framework.

Pesquisa de natureza aplicada, uma vez que será gerado no final um mapeamento completo sobre o problema, que resultará no desenvolvimento de um framework de trabalho e suas respectivas ferramentas. Possui caráter exploratório, que segundo Cervo, Bervian e Silva [8] estabelecem critérios, métodos e técnicas para a elaboração de uma pesquisa e visa oferecer informações sobre o objeto desta e orientar a formulação de hipóteses, envolverá levantamento bibliográfico; análise de similares; levantamento e análise de dados que estimulem a compreensão.

Posteriormente os conhecimentos adquiridos foram aplicados na produção do framework com detalhamento de todas as etapas de desenvolvimento. A partir da primeira versão da ferramenta, foi realizada uma investigação qualitativa, por meio de testes e avaliações a fim de comprovar a consistência do mesmo e corrigir falhas para implementação dentro do laboratório GAIA e disponibilização da ferramenta para a comunidade.

Das etapas de desenvolvimento tem-se:

- Levantamento bibliográfico das metodologias Agile, Design Thinking, desafios e motivações no desenvolvimento de software;
- Levantamento de dados: análise de similares no mercado, buscar frameworks existentes de grandes empresas;
- Estudo de caso com projetos realizados no laboratório de software GAIA e sua dinâmica de trabalho;
- Levantamento de dados: questionários com profissionais, acerca dos desafios de concatenar design e programação que indiquem a melhor forma de construir ou readequar um framework;
- Criação da proposta de framework e suas respectivas ferramentas para auxiliar o desenvolvimento de software;

3 FUNDAMENTAÇÃO TEÓRICA

3.1 Engenharia de Software

Assim como em muitas outras disciplinas, conceituar uma área do conhecimento, neste caso Engenharia de Software, não é uma tarefa simples, porém a fins de abordar a mesma torna-se necessário partir de um ponto comum e talvez generalizado do que se entende por Engenharia de Software, tendo em mente que os processos contidos na teoria variam de acordo com o projeto e software a ser desenvolvido.

A engenharia de software é uma disciplina que rege todas as etapas em um processo de desenvolvimento de software, de modo a garantir a eficácia dos processos e excelência nos resultados. De acordo com Laplante [32] engenharia de software pode ser definida como uma abordagem sistemática de análise, design (desenvolvimento), avaliação, implementação, teste e manutenção de software. Na abordagem de engenharia de software, diversos modelos para o ciclo de criação de um software são definidos, e várias metodologias são abordadas para análise e avaliação das diferentes fases de um modelo de criação.

Laplante [32] aponta que, além disso, a engenharia de software inclui também atividades relacionadas a todo artefato ligado à própria engenharia de software como ferramentas e documentação. Para complementar esta definição o autor destaca duas principais atividades relacionadas com a área que resume as funções executadas, que são:

- Modelar: diz respeito a uma atividade de tradução. O conceito de um software é traduzido em uma série de requisitos, que em sequência é convertido em um design, que é convertido em código para então ser traduzido em compiladores e montadores, finalmente chegando a sua forma executável por uma máquina;
- Otimização: diz respeito a especificar o método mais eficiente, claro e desejável de modelagem, ou seja, estudo do processo de construção em si, com foco nas etapas e não no resultado.

Cada uma destas etapas envolve erros e o papel do engenheiro é reduzi-los por meio da aplicação de princípios adequados a tarefa. Diversos autores atentam para a "crise do software" que ocorre desde a origem do mesmo, fato que diz respeito aos complexos desafios impostos aos profissionais da área e sentimento de desamparo

teórico e prático que guie as produções de modo a mitigar os riscos.

Wazlawick [67] os riscos mais comuns que envolvem um projeto de software são:

- Estourar cronograma;
- Estourar orçamento;
- Produto que não atende as expectativas do mercado e de baixa qualidade;
- Produtos não gerenciáveis, difíceis de manter e sem chances de evoluir.

O engenheiro de software de software, portanto tem o metapapel [67] de fornecer a equipe de desenvolvedores as ferramentas e processos que deverão ser utilizados e realizar a constante verificação se o uso destes estão sendo feitos e forma efetiva e otimizada. Garantindo também a melhoria contínua dos processos que viabiliza a produção.

Pressman [50] em um capítulo do seu livro Software Engineering A Practitioners Approach mostra de forma bastante interessante alguns mitos que permeiam o projeto de desenvolvimento de software, que serão resumidos a seguir:

- Ter um manual de padrões e processos definidos e fixos não é suficiente para gerenciar uma equipe inserida em um cenário onde práticas e posicionamento precisam ser revistos e alinhados constantemente;
- Equipamentos e ferramentas de última geração não são sinônimos de resultados de alta qualidade. Possuir bons equipamentos é apenas um da grande quantidade de requisitos para sucesso de um projeto;
- Adicionar mais profissionais em um projeto que estourou o cronograma não irá melhorar o desempenho do mesmo, podendo até mesmo gerar atraso ainda maior;
- Terceirizar processo de desenvolvimento não significa maiores chances de sucesso, caso não haja forte acompanhamento e gerência de tempo e recursos investidos;
- Uma ideia generalizada não é suficiente para iniciar um projeto. Qualquer projeto precisa ser delimitado de forma sistemática para que os requisitos iniciais sejam mais próximos à realidade do produto no final;
- Software em si não são flexíveis a mudanças de última hora se o modelo de processo de desenvolvimento não estiver preparado para tais ocorrências;
- Para testar ou analisar a qualidade de um programa, não é necessário aguardar uma instância operacional do mesmo. As etapas de análise podem ser realizadas desde início do projeto, na forma de análise formal técnica.

A produção de software é uma atividade de resolução de problemas que é

alcançada por meio de modelagem. Como uma solução de problemas, engenharia de software é uma atividade humana que é afetada por experiências prévias e está altamente sujeita a erro humano. Portanto, o engenheiro de software deve reconhecer e tentar eliminar esses erros.

3.1.1 Controle de Qualidade

Na engenharia de software, a qualidade do software está relacionada a dois conjuntos de fatores. O primeiro diz respeito à *função*, ou seja, o software deve estar em conformidade com as especificações definidas pelo que o usuário espera que o software execute. O segundo é o requisito não funcional, que é definido por como o usuário espera que o software seja executado. Exemplos de requisitos não funcionais são a disponibilidade, confiabilidade, eficiência e usabilidade do software.

O controle de qualidade de um software de acordo com McManus e Wood-Harper [39] relaciona-se com a análise de todo o processo. Todas as atividades que forem executadas devem ser claramente evidenciadas e definidas, incluindo a ordem em que devem ser realizadas e quando estas são consideradas completas. Para garantir melhoria de processo, este precisa ser usado em diversos contextos, pois quando o mesmo processo é usado em diferentes projetos, torna-se possível encontrar maneiras pelas quais o processo pode ser melhorado. Tais melhorias são geralmente pequenas e relativamente fáceis de implementar, que ao longo do tempo levam a economias significativas e têm um benefício financeiro positivo. O terceiro conceito é o de métricas, que devem ser coletadas e documentadas para determinar se mudanças incorporadas no processo, são realmente melhorias.

Em termos de qualidade McManus e Wood-Harper [39] sugere que qualquer tipo de produto a ser desenvolvido apresenta suas próprias demandas de qualidade, entretanto softwares são particularmente complicados em relação à qualidade pelos seguintes motivos:

- Softwares não possuem existência física, são intangíveis;
- A falta de conhecimento das necessidades do cliente no início de cada projeto;
- As mudanças bruscas das necessidades do cliente ao longo do desenvolvimento;
- O desalinhamento da evolução entre hardware e software, visto que são interligados;

- As altas expectativas dos usuários, principalmente ao que diz respeito a adaptabilidade do sistema.

Para garantir a qualidade e analisar os processos, García et al. [22] aponta que ao longo das últimas décadas, no campo da engenharia de software, a atenção tem sido focada em melhorar os processos de software usando padrões como CMMI, incorporando desenvolvimento ágil (principalmente XP e SCRUM), ou promovendo a melhoria da qualidade do produto (ISO 25010).

3.1.2 Iterações

Antes de abordar as metodologias ágeis é necessário definir e exemplificar o conceito de iteração, uma vez que este é a força motriz por detrás de metodologias Agile. Por definição semântica Iteração, de acordo com dicionário Michaelis [41] pode ser definida como: ação de Iterar ou de Repetir. Procedimento de resultado de uma equação, por meio de sucessivos cálculos, em que o *objeto dessa* é o *produto* daquela que *antecede*.

Em metodologias mais tradicionais o processo de desenvolvimento de software percorre individualmente e em sequência cada disciplina envolvida, de forma isolada. De acordo com Iterar [28] os métodos tradicionais (figura 1), como o cascata, resultam em um acúmulo de integração tardia na implantação, quando, pela primeira vez, o produto é criado e o teste começa. Aparecem os problemas que permaneceram ocultos por todo o processo de Análise, Design e Implementação, e o projeto é paralisado enquanto começa um longo ciclo de correção de erros.

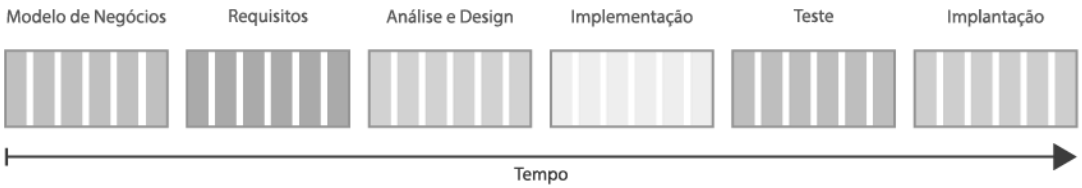

Figura 1: Modelo tradicional

Nos modelos iterativo e incremental (figura 2) um projeto é dividido em um subconjunto de funcionalidades. Bezerra [4] cita um exemplo: um projeto de um ano pode ser dividido em iterações de três meses, no qual cada iteração teria como escopo ¼ dos requisitos passando por todo o ciclo de vida do software: análise, projeto, código e teste. Deste modo no final da primeira iteração tem-se um sistema que executa ¼ das funcionalidades necessárias.

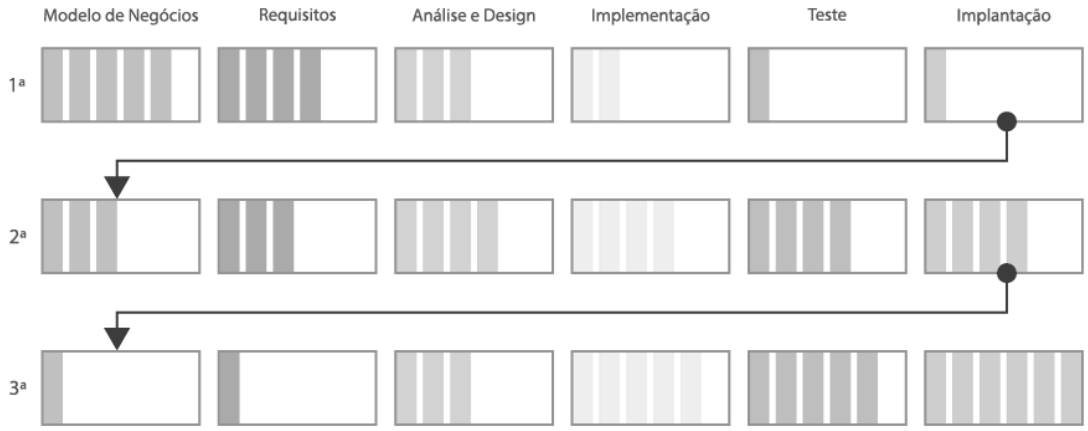

Figura 2: Modelo iterativo e incremental

Torna-se pertinente diferenciar aqui desenvolvimento Iterativo e Incremental. De acordo com Gordon & Gordon [23] método incremental é aquele em que o produto é construído e entregue em pedaços, que representam um grupo de funcionalidades completas. Já o método iterativo tem seu progresso por meio de liberações sucessivas de pequenas partes (pontos críticos) de um software para que este seja executável, realizando testes ao final de cada iteração para correção de *bugs* e refinamentos futuros.

Segundo IBM [26] as iterações ajudam a reduzir os riscos envolvidos em um projeto de software e entregar de forma antecipada funcionalidade ao usuário. O projeto em método iterativo é dividido em subconjunto de funções que devem ser entregues ao fim de cada iteração para que possam ser testadas pelo cliente/usuário e gerado *feedback* a partir dos resultados. Este método se baseia em um loop de feedback e por isto é altamente receptivo a mudanças.

3.1.2.1 Ciclo de vida: cascata

Modelo cascata, ou modelo sequencial linear, segundo Nakagawa [42] é o modelo mais antigo e mais amplamente usado pela engenharia de software convencional e requer abordagem sistemática onde o resultado de uma fase constitui o início da outra (Figura 3). Cada etapa do processo de desenvolvimento é abordada em sua totalidade e de forma isolada, ou seja, a etapa de Projeto e Design só se inicia quando as etapas de coleta e definição de requisitos estejam finalizadas.

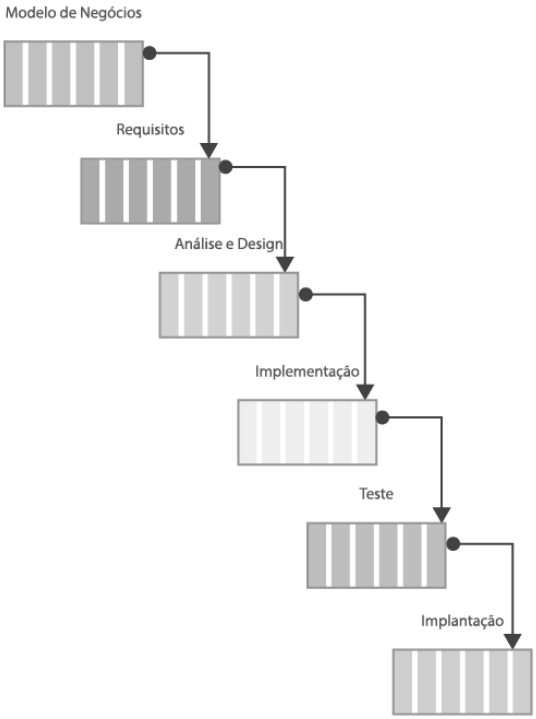

Figura 3: Método cascata

Medeiros [40] aponta que o modelo cascata é vantajoso quando os requisitos do sistema são amplamente conhecidos e/ou quando um sistema já consolidado precisa ser aperfeiçoado e/ou requer a adição de novas funcionalidades. O autor segue apontando as desvantagens de tal modelo no atual cenário de Engenharia de Software:

- Projetos reais dificilmente seguem fluxo linear no que diz respeito às exigências e mudanças bruscas de mercado. Mesmo que iterações possam ser identificadas e apresentam uma relativa maleabilidade, o modelo como um todo não aceita mudanças. Portanto a não ser que cada etapa consiga consolidar-se livre de erros, o processo ficará comprometido e a fase de correções será tão longa quanto o processo inicial de construção;
- O cliente não tem acesso rápido a um sistema operacional, uma vez que este só existirá no final de todas as iterações, e quando o cliente acessar o sistema, caso os requisitos iniciais não estiverem alinhados com suas expectativas, o resultado pode ser desastroso;
- Bloqueio no fluxo de produção entre membros da equipe, pois precisam esperar que outros completem suas tarefas para dar sequência no trabalho. Levando a uma queda de produtividade e desperdício de recursos financeiros.

Apesar dos problemas mencionados Bezerra [4] aponta para o fato de que este

modelo foi o mais utilizado durante anos desde início da computação até recentemente. Porém devido ao aumento da complexidade dos sistemas e da lógica de mercado ciclos mais receptivos a mudanças e que lidam de forma eficiente com erros, como incremental e iterativo são mais utilizados.

3.1.2.2 Ciclo de vida: espiral

Originalmente proposto em 1986 o modelo espiral é fortemente voltado à redução de riscos envolvidos no desenvolvimento de um projeto. Wazlawick [67] explica que o projeto neste caso é fragmentado em subprojetos, ou iterações, onde cada fragmento aborda um ou mais elementos de alto risco, de forma cíclica até que todos os riscos sejam identificados e tratados.

De acordo com Nakagawa [42] este modelo integra a natureza iterativa de prototipação de software em um ambiente controlado e sistemático do modelo cascata. A divisão das atividades de cada etapa são chamadas de região de tarefas, que no modelo mais abordado possui 4 regiões, mas que podem variar de três a oito.

Na figura 4 temos o modelo exemplificado da espiral de desenvolvimento onde cada volta dada representa uma fase do sistema, que não são pré-determinadas (somente exemplos na imagem) no início do projeto as fases são delimitadas pela equipe e podem sofrer alterações ao longo do processo.

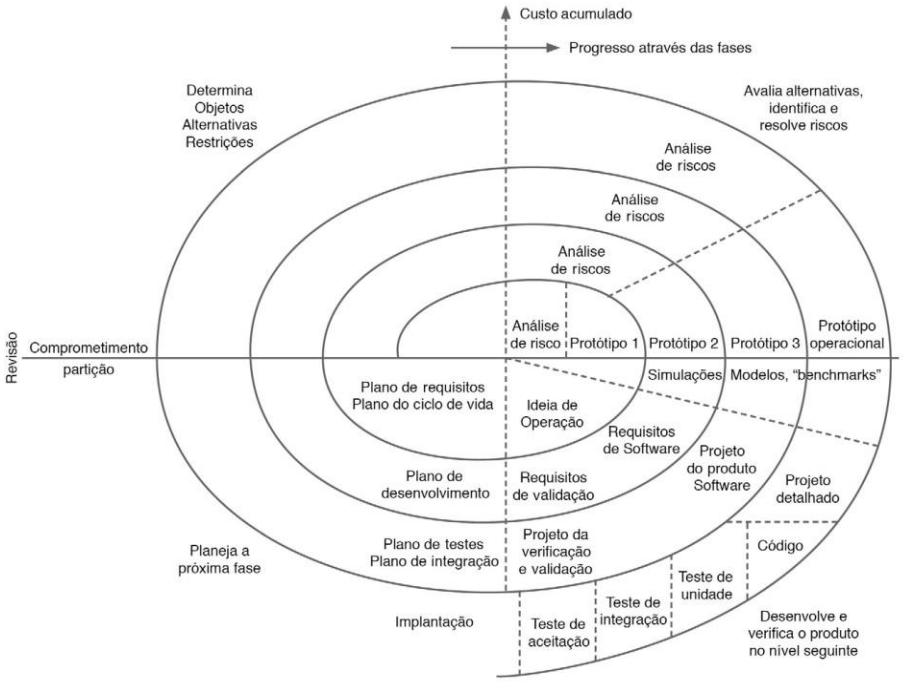

Figura 4: Modelo espiral [67]

Wazlawick [67] aponta para as seguintes vantagens deste modelo:

- As primeiras iterações apresentam custo reduzido no que diz respeito a tempo e recursos, e também são onde os problemas mais críticos são resolvidos;
- As iterações seguintes à alocação de tempo e recurso são maiores, porém os riscos são drasticamente reduzidos, o que é altamente desejável em projetos de grande porte;
- Após os potenciais problemas terem sido identificados, a equipe pode passar a trabalhar em um ciclo final semelhante ao linear de cascata.

Apesar de vantajoso o autor aponta para o fato que este é um modelo altamente complexo e requer uma gerência eficiente em total controle do processo. E que só se beneficia deste modelo projetos grandes com alto risco onde os requisitos são poucos conhecidos e grau de originalidade elevado, como o desenvolvimento jogos eletrônicos.

3.1.2.3 Ciclo de vida: incremental e evolutivo

Segundo Bezerra [4] processo incremental relaciona-se com a ideia de aumentar pouco a pouco a área de um sistema. O objetivo é trabalhar junto com o usuário de maneira incremental até que o produto final seja obtido [42]. O modelo incremental se mostra particularmente importante quando não se é possível delimitar o escopo de requisitos iniciais de forma completa, e apresenta resultados positivos quando aplicado em sistemas pequenos, onde a quantidade de informação e consequentemente a de erros e riscos são drasticamente reduzidas.

De acordo com Iterar [28] os tipos de iteração recorrentes neste tipo de ciclo de vida são:

- Uma iteração curta para definição do case de negócio junto com escopo e visão;
- Iteração de elaboração para criação de uma base de arquitetura estável, como ponto de partida;
- Iteração de construção aonde o loop de feedback entre cliente e equipe gera aprimoramento do software;
- Diversas iterações de transição para migrar o software aos usuários.

No desenvolvimento de qualquer produto comercial, o que normalmente se aplica aos softwares, a tarefa de criação estende-se por meses, senão anos até sua conclusão que muitas não implicam no fim do desenvolvimento, mas no início de toda uma nova etapa de manutenção e adequação às constantes evoluções no mercado

tecnológico. Portanto Bezerra [4] aponta que se torna intuitivo fragmentar o trabalho em porções tangíveis (iterações) tendo ao fim de cada uma como resultado um incremento ao produto inicial (incremental), deste modo à equipe volta-se ao constante refinamento dos resultados e processos.

3.1.3 Desenvolvimento Ágil - Agile

Com a constante evolução do mercado de software e o aumento pela demanda em inovação é necessário às empresas se posicionarem estrategicamente como geradoras de inovação. Para que isso ocorra é preciso que métodos tradicionais e isolados sejam deixados para trás. Á partir da revisão sistemática realizada em conjunto com esta pesquisa pode-se verificar nos estudos a preferência por uma abordagem mais técnica pautada em pontos específicos da produção de software e também a necessidade de novas teorias de *convergência* de metodologias mais compatíveis com a realidade multidisciplinar das equipes e empresas e software.

Deste modo empresas optam por adotar metodologias de desenvolvimento mais condizentes com um mercado propenso a mudanças e riscos, entram aqui as metodologias ágeis. Segundo Sommerville [61] estas metodologias se destacam pela proposta de criação softwares de forma *iterativa* nas quais os requerimentos, o processo, o desenvolvimento e testes são intercalados, ou seja, o software vai sendo disponibilizado de forma integral aos pontos críticos, mas em séries de incrementos ao longo do processo.

Além da adoção do processo iterativo, o desenvolvimento ágil se baseia nos 12 princípios estabelecidos no manifesto ágil [51], estes são:

- A maior prioridade é satisfazer o cliente, por meio de entrega antecipada e contínua de um software consistente;
- Mudanças são sempre bem-vindas, até tarde em desenvolvimento. Processos ágeis aproveitam a mudança para vantagem competitiva;
- Entregar software operacional frequentemente, semanalmente ou mensalmente, com uma preferência à escala de tempo mais curta;
- Clientes e desenvolvedores devem trabalhar juntos ao longo do projeto;
- Construa projetos em torno de indivíduos motivados. Dê-lhes o ambiente e o apoio adequado, e confie neles para fazer o trabalho;

- O método mais eficiente e eficaz de transmitindo informações para uma equipe de desenvolvimento é uma conversa cara-a-cara;
- O software operacional (que funciona) é a principal medida de progresso;
- Os processos ágeis promovem o desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente;
- Atenção contínua a excelência técnica e um *bom design* aumenta a agilidade;
- Simplicidade - a arte de maximizar o valor de que não precisou ser feito é essencial;
- As melhores arquiteturas, requisitos e projetos emergem de equipes que se auto-gerenciam;
- Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz, para melhorar e ajustar o seu comportamento.

Em um desenvolvimento ágil, a priorização é importante, pois ajuda a restringir o foco do profissional a uma quantidade limitada de itens específicos. O método ágil procura sempre o resultado mínimo viável para o lançamento - o que o designer e o consumidor alvo precisam agora - sabendo que podem e irão incrementá-lo posteriormente.

Em um estudo de caso realiado por Hamdan e Alramouni [24] foi analisado atributos de qualidade de software em frameworks tradicionais e frameworks que adotaram desenvolvimento ágil. Na tabela 1 abaixo podemos constatar os benefícios de adoção de tal metodologia e o seu impacto no processo produtivo.

Tabela 1: Qualidade de software: framework tradicional x ágil [24]

Critério de Qualidade	**Método Tradicional**	**Desenvolvimento Ágil**
Tempo de Desenvolvimento	Desenvolvedores trabalham na produção de todos os requerimentos	Desenvolvedores trabalham em um único requerimento
Bugs Encontrados	Como os testes são realizados após grande porção do software é finalizada, são encontrados quantidade grande de erros e bugs.	Cada recurso é testado e consertado após ser finalizado. Redução na quantidade de erros.
Tempo para Entrega	Após finalizar todos os requerimentos.	Após finalizar um requerimento.
Aplicação Teste	Testes são feitos somente no final.	Pequenas partes são testadas e resultados são enviados imediatamente para os desenvolvedores.

Tabela 1: Continuação da página anterior

Documentação	O resultado é documentado apropriadamente. As especificações são documentadas antes do início de produção. O trabalho de produção é documento após ser finalizado.	Os requerimentos são documentados. Ferramentas automáticas geram dados e estatísticas sobre os recursos desenvolvidos, resultados de testes e bugs encontrados.
Gerenciamento de Mudanças	Mudanças são aceitas após longo processo e aprovação. Mudanças são integradas por meio de patch (compactado grande de modificações).	Mudanças são aceitas ao longo do processo, e são adicionadas como novos requisitos pelo cliente.
Modelo de Custo	O tempo e esforço de fazer manualmente a construção e integração de um software.	Custo de ter ferramentas e servidores para desenvolvimento ágil.

Nos frameworks propostos, desafios presentes na metodologia ágil, como falta de pesquisa preliminar e observação do problema são preenchidas pelo caráter exploratório e convergente do design thinking, quebrando o trabalho em iterações que são avaliadas a cada passo para busca de possíveis soluções.

3.1.3.1 Scrum

Scrum segundo Laplante [32] é uma metodologia fortemente pautada no manifesto ágil, onde a comunicação verbal de todos os membros da equipe e em todas as partes interessadas é incentivada. O princípio fundamental do Scrum é que as abordagens tradicionais da solução de definição de problema nem sempre funcionam, e que um processo de descoberta formalizada torna-se necessário.

As principais características do Scrum de acordo com Schimiguel [56] são:

- É um processo ágil para gerenciar e controlar o desenvolvimento de projetos;
- É uma base para outras práticas de engenharia de software;
- É um processo que controla o caos resultante de necessidades e interesses conflitantes;
- É uma forma de aumentar a comunicação e maximizar a cooperação;
- É uma forma de detectar e remover qualquer impedimento que atrapalhe o desenvolvimento de um produto;
- É escalável desde projetos pequenos até grandes projetos em toda empresa.

Quando se prática SCRUM é recorrente utilizar o quadro de tarefas Scrumm (figura 5) para organizar de forma visual o backlog das iterações, ativas ou não, e planejar ações futuras.

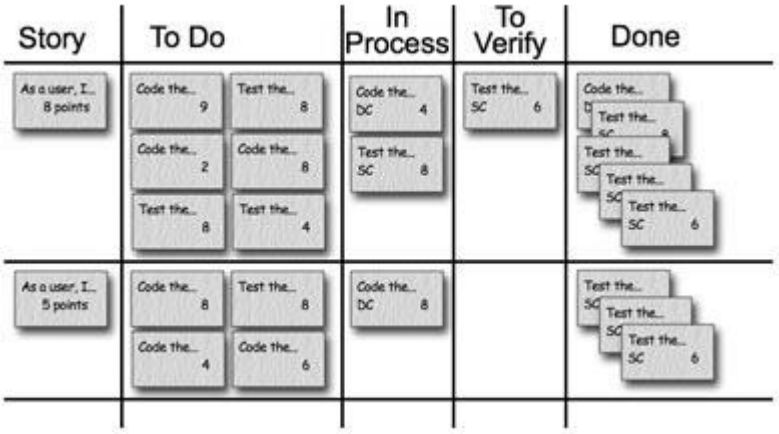

Figura 5: Scrum task board [57]

3.1.3.2 Extreme programming

Extreme Programming ou XP é uma das metodologias ágeis mais utilizadas, geralmente adotada por equipes pequenas e requer relativamente uma reduzida alocação de recursos. De acordo com Laplante [32] enquanto em modelos iterativos e evolucionários ainda é possível delimitar as fases de forma similar ao modelo cascata, no XP após a fase inicial de análise todas as atividades acontecem mais ou menos de forma contínua ao longo do ciclo de vida (Figura 6).

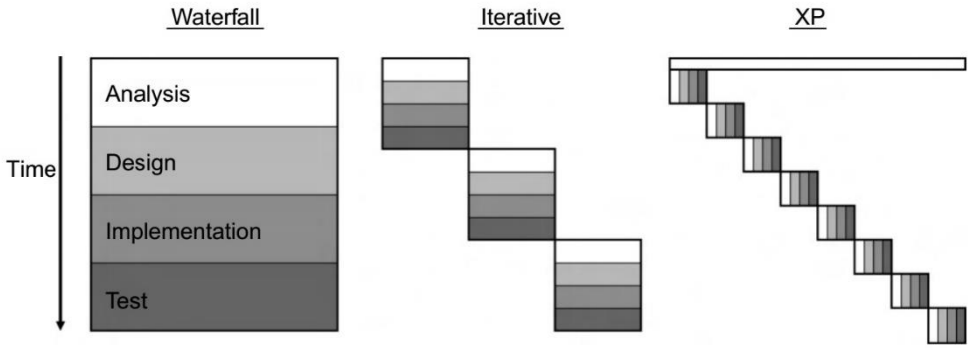

Figura 6: eXtreme programming [32]

Assim como o Scrum o XP também é totalmente pautado no manifesto ágil e apresenta etapas e características muito semelhantes. A diferença notável entre os dois talvez esteja na maior abrangência de área do Scrum, enquanto o XP é totalmente

pautado no desenvolvimento de software.

3.1.4 Desafios em Projetos de Desenvolvimento de Software

Pessoas são o ponto fundamental e crítico para o desenvolvimento de software, uma vez que um programa é uma entidade humana feita para ser usada por pessoas e, além disso, é produto criativo destas. Acuña et al. [1] demonstra que a maioria das falhas em desenvolver um software é causa direta de fatores humanos. Ao abordar traços de personalidade é importante ressaltar que os trabalhos consultados se referem a um conjunto de atributos e atitudes que uma *equipe* possui perante a um projeto e como isso influencia os resultados e a produtividade, ou seja, não se tratam de características pessoais e subjetivas de cada pessoa (que pode até ser estudado, porém não cabe a este tipo de pesquisa ou área), muito menos das filosofias de vida individuais ao encarar problemas cotidianos.

Em equipes que adotam desenvolvimento ágil Yilmaz et al. [70] em uma extensa pesquisa acerca de traços de personalidade em equipes de software, aponta que a natureza altamente social do desenvolvimento ágil é compatível com traços como abertura para novas experiências que é caracterizada como possuir boa comunicação e capacidade de inovação. Para haver produtividade em equipes ágeis é necessário disciplina em relação as habilidades organizacionais, abertura e adaptabilidade para mudanças e críticas.

Em uma revisão sistemática por Steinmacher et al. [62], que analisou 49 estudos pertinentes ao caso, acerca das dificuldades encontradas por integrantes novos em equipes de software, que também vale para equipes novas que se formaram, ou no caso de um laboratório de software, equipes temporárias, foram encontrados quatro principais obstáculos, são:

- *Falta de interação social entre os membros da equipe:* o que os membros da equipe devem atentar é construir um processo pautado na comunicação e interação. Embora cada indivíduo deva trabalhar para construir sua própria identidade e voz dentro do projeto, as decisões e propostas devem ser pautadas e construídas na opnião conjunta, ou seja, não só uma voz deve expressar uma ideia;
- *Receber respostas atrasadas à dúvidas ou questões:* falta de comprometimento e consideração ao requerimento do outro. Foi relatado que em equipes que trabalham a distância, a média de espera para resposta é de 48 horas. Isto resulta

não só no atraso do projeto, mas também em uma queda de motivação da equipe toda;

- *Falta de experiência técnica:* falta de experiência ou expertise técnica para solução direta de problemas é uma das grandes causas de falhas em equipes. O autor afirma que a formação acadêmica não está diretamente conectada com boa experiência técnica, e enfatiza a importância das habilidades e experiências técnicas serem verbalizadas e compartilhadas entre a equipe, desta forma direcionando foco a preencher estas lacunas;
- *Dificuldade de localizar tarefa adequada para se começar um projeto:* diz respeito a dificuldade dos membros da equipe em localizar as tarefas em que podem contribuir baseada em suas habilidades técnicas. Problema relacionado ao item anterior que enfatiza a importância do time expor as experiências e expertises de cada um, e atentarem depois para a distribuição adequada de tarefas.

Além destes os autores citam também falta de compreensão das reais necessidades do projeto, falta de Ambiente virtual para exposição de ideias, falta de habilidade técnica por parte da equipe e inexistência de metodologia bem estruturada para gerenciamento do projeto.

3.2 Gestão de Design

Segundo Norman e Jerrard [46] ambientes organizacionais que fazem uso de profissionais do design mas não possuem gerenciamento voltado para tal área, impõe a tais profissionais um abismo entre eles e o resto da organização pois o design fica sem meios de se relacionar com a estratégia corporacional e comunicar-se efetivamente com outras disciplinas.

A Gestão de Design, portanto vem sendo amplamente discutida como um fator chave para o sucesso corporativo, como pode ser visto em estudos internacionais de autores [55][12] que quanto mais uma empresa de posiciona no mercado como geradora de inovação, maior o papel do design e seu grau de importância em decisões relacionadas ao mesmo; assim a empresa que em retorno ao investimento e dependência de design interno ganha vantagem competitiva sustentável. Deste modo como apontado Ward, Runcie e Morris [66] verifica-se assim o motivo do interesse crescente das empresas de estruturar processos de design internos que atendam suas demandas internas sem a necessidade de terceirização por acarretar em custos e déficit na qualidade dos processos.

De modo geral a Gestão de Design [35] é a implantação das metodologias de design em qualquer programa ou quadro de atividades de uma organização, alinhando as estratégias de design com os objetivos de curto e longo prazo da empresa, realizando a coordenação de recursos em todos os níveis de atividade, seguindo este pensamento Kumar [31]aponta que o desafio para as empresas não está só em adotarem métodos de inovação de design, mas também fundi-los com os processos já existentes de modelos de negócio e desenvolvimento de tecnologia.

Criar um software desde o conceito requer o balanço entre lucro para empresa e satisfação do consumidor. Desta forma designers desempenham um papel importante como integrador das contribuições das diferentes expertises na melhor oferta possível de resultados. Para Sato [55] possuir um framework de trabalho que integre o design em sua raiz é garantia que os investimentos em inovação estejam sempre alinhados aos resultados e objetivos da empresa, uma vez que o design aborda desde problemas estratégicos até auxilio no processo criativo de um software.

Tocante a este assunto Casas e Merino [7] apontam que a gestão de design tem seu foco voltado a projetos individuais de design e com evolução e melhorias incrementais, e de complemento a isso o design thinking, que será discutido a seguir, se apresenta neste cenário como uma mudança mais radical na maneira de uma organização fazer negócios.

3.2.1 Design Thinking

Apresentado por Tim Brown, diretor Executivo da IDEO o termo Design Thinking consiste em uma nova abordagem para resolução de problemas não só ligados à indústria criativa, mas a diversos campos e experiências. De acordo com Brown [5], "design thinking é em sua essência a capacidade do pensamento integrativo". O design thinking se difere do pensamento em grupo, mas pode ocorrer em grupos, buscando liberar a criatividade dos envolvidos no projeto.

A confusão entre o termo design e design thinking (DT), assim como DT e Gestão de Design são tópicos de discussão recorrentes no cenário acadêmico. Porém é importante partirmos de um ponto comum, mesmo que genérico do que está sendo tratado neste trabalho. Na tabela 2 foram elencadas por Rolim [52] a relação e divergências que podem ser levantadas entre os dois termos.

Tabela 2: Design vs design thinking [52]

DESIGN	**DESIGN THINKING**
Formato: Criar e Idealizar o visual de itens gerais.	Descobrir oportunidades para solução de problemas
Função: Concepção e desenvolvimento de novas "utilidades".	Redefinir problemas para encontrar oportunidades de inovação
Requisitos: Treinamento e Desenvolvimento de habilidades específicas para ser Designer.	"Qualquer um" poder ser design thinker.
Foco: Individuo e Produto	Times e Experiências

Bartolomeu [3] ressalta que não há apenas um modelo definido para design thinking, havendo diversas propostas que podem ser aplicadas em diferentes dimensões de um projeto/organização. O modelo clássico é o proposto pelo próprio Brown e talvez seja um dos modelos mais conhecidos. Brown [5] descreve três elementos essenciais para qualquer design de sucesso, sendo eles:

- *Insight:* Compreensão repentina de um problema, ocasionada por uma percepção mental clara e, geralmente intuitiva, dos elementos que levam a sua resolução. Os insights são gerados a partir da observação de experiências verdadeiras que o ser humano vivencia em seu cotidiano, de seus atos impensados, direcionando o design thinker as necessidades não atendidas do público em observação. Essa fase de observação e pesquisa de campo contribui demasiadamente para o lançamento de um projeto, sendo altamente crítica;
- *Observação:* A observação constitui-se em um trabalho de campo, onde as pessoas são observadas em tudo o que fazem e não fazem, assim como o que ouvem e dizem e o que não dizem. Esta por sua vez, se baseia na qualidade e não em dados quantitativos. De acordo com Brown [5] para alcançarmos insights que nos ensinem algo novo e surpreendente, "precisamos nos voltar aos extremos, aos locais em que esperamos encontrar usuários radicais, que vivem de forma diferenciada, pensam de forma diferenciada e consomem de forma diferenciada".
- *Empatia:* Segundo Brown, empatia é criar uma conexão com as pessoas que estão sendo observadas em nível fundamental. O autor enfatiza a missão do design thinking que é traduzir observações em insights, e estes em produtos e serviços para melhorar a vida das pessoas.

Para o autor, DT é uma abordagem a solução de problemas onde inspiração,

ideação e implementação não ocorrem de maneira linear, mas como um sistema de espaços que se convergem (figura 7), até que as ideias da equipe sejam lapidadas, e explorem-se novas possibilidades. Para tanto, o design thinking caracteriza-se por um processo exploratório, não linear, que promove a interação durante a execução de um projeto.

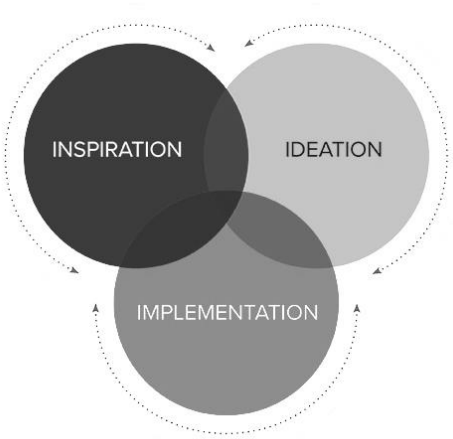

Figura 7: IDEO design thinking [47]

Outro modelo bastante conhecido é o de Lindberg [34] da escola de Stanford (figura 8). Este modelo foi construído a partir do modelo proposto pela IDEO, difere no que diz respeito à definição forte de etapas em prol da sequencialidade de um projeto sem perder a flexibilidade dos espaços convergentes propostos por Brown. Neste modelo tem-se nas duas primeiras etapas são dedicadas a análise do problema, seguidos da sistematização de informações, para enfim chegar a área de desenvolvimento e testes.

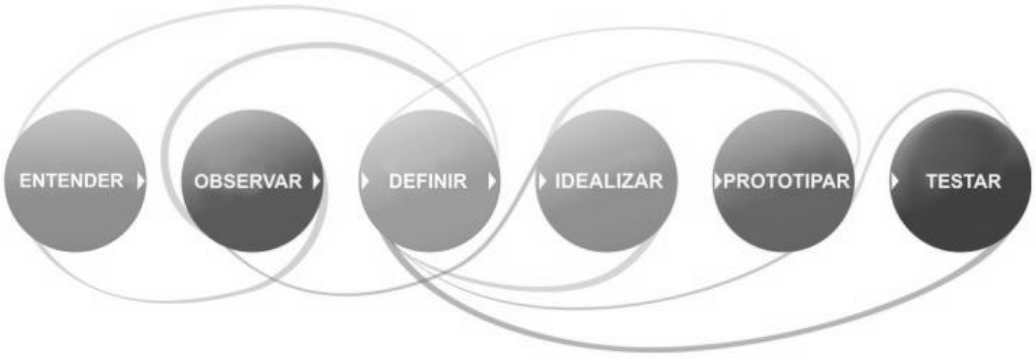

Figura 8: Stanford design thinking [34]

Assim como o design se depara com limitações durante o processo de desenvolvimento do produto, o design thinking não se difere nesse ponto. Para Brown [5] “a disposição e até a aceitação empolgada das restrições constituem o fundamento

do design thinking". Tais restrições estão relacionadas a três critérios que não devem ser apenas solucionados, mas necessitam estar em equilíbrio e harmonia, sendo eles:

- Praticabilidade: o que é funcionalmente possível num futuro próximo;
- Viabilidade: o que provavelmente se tornará parte de um modelo de negócios sustentável;
- Desejabilidade: o que faz sentido para as pessoas.

Concluindo, Bartolomeu [3] aponta que apesar dos diversos modelos, as metodologias de Design Thinking sempre mantem em seu núcleo o foco do processo no ser humano, a empatia como ferramenta essencial para avaliação e solução de problemas e o forte fomento a criatividade.

3.2.2 Ferramentas de Design Thinking

As ferramentas de design thinking, são na verdade o detalhamento dos diversos processos que ocorrem nas fases já citadas anteriormente. Elas podem ser vistas como exercícios e atividades no decorrer de um projeto para potencializar empatia, criatividade e centrar todo o desenvolvimento no ser humano. Em processo de TI que possuem alto foco do desenvolvimento técnico, o Design thinking entra como contrapeso para equilibrar a equação.

As ferramentas citadas a seguir é um compilado retirado do livro Design Thinking Inovação em Negócio de Vianna et al. [65] que por si também é um compilado de diversos autores e modelos de DT, apresentando cases reais de como essas ferramentas foram aplicadas. Demais citações complementares, que forem julgadas interessantes, serão devidamente referenciadas.

3.2.2.1 Imersão

Reenquadramento: analisar questões projetuais sobre novas perspectivas, desconstrução de paradigmas estabelecidos e suposições com foco em soluções inovadoras. Inicia-se com coleta de dados acerca do problema, com participação ativas dos clientes e usuários finais, com exercícios de analogia e outras dinâmicas que incentivam olhar diferenciado do cenário proposto. Os dados são então organizados utilizando ferramentas como mapa mental, jornadas, personas para identificar questões não levantadas e escolher as táticas para o próximo ciclo de desenvolvimento.

Pesquisa Exploratória: como nome já diz é a realização de uma pesquisa de

campo para levantamento de dados e insumos que colaborarão para o desenvolvimento de perfis de usuários, ambientes, ciclo de vida do produto ou serviço que serão desenvolvidos em técnicas de imersão mais aprofundadas. Serve para familiarizar a equipe com a realidade de uso do produto/serviço assim como os atores envolvidos e suas demandas/necessidades que serão utilizadas como guia para definição de *requisitos*.

Pesquisa Desk: como nome já diz é uma pesquisa que busca informações complementares que não podem ser encontradas em uma pesquisa de campo, como por exemplo, websites, livros, artigos e blogs. Pode acontecer de ser utilizada esta ferramenta ao longo de todo um projeto no auxílio de questões e obstáculos que surgem e precisam ser aprofundados. As informações podem ser registradas em cartões de insight e organizados em uma arvores ou mapa de temas, para o cruzamento de dados que permitem a identificação de *gaps* a serem explorados dentro do projeto. Fonseca [18] aponta que não se deve confundir esta ferramenta com Benchmarking, pois o alvo desta é somente um processo contínuo de comparação de produtos, serviços e práticas.

3.2.2.2 Imersão em profundidade

Consiste em criar um perfil mais aprofundado dos atores e cenários envolvidos no desenvolvimento de um projeto. Com foco altamente humano que busca salientar questões como modo de pensar, afetos e atitudes de um grupo perante um serviço/produto. Para isso têm-se as seguintes ferramentas.

Entrevistas: podem ser por meio de uma conversa direta ou questionários que buscam obter informações por meio de perguntas, exemplos e outros métodos. Mais do que entender o meio ou como um indivíduo toma determinadas atitudes, a entrevista procura alcançar os *porquês* por detrás das ações, incentivando a busca de significados para expandir entendimento sobre comportamento, mapear padrões e peculiaridades assim como exceções a regras.

Cadernos de Sensibilização: utilizado para obter informações de forma indireta dos usuários/clientes, por meio de atividades propostas pelo designer que instiguem o usuário a relatar seu universo e atividades. Este tipo de informação é bastante relevante durante a fase de imersão, pois possibilita entendimento profundo do universo do cliente. As atividades propostas podem ser relatos escritos das atividades desenvolvidas, registros fotográficos, colagens, etc.

Sessões Generativas: Expansão da ferramenta anterior, trazendo para o coletivo as atividades de reflexão propostas anteriormente. No formato de grupo focal, os participantes realizam tarefas focadas a estimular troca de experiências. Este tipo de ferramenta permite ao designer observação mais completa das atividades e relacionamento dos usuários e informações.

Um dia na vida: trata-se de uma simulação de como seria um dia na vida do usuário, onde o designer se coloca no papel deste. Este tipo de exercício é altamente recomendado para equipes que não conseguiram criar empatia necessária com um projeto e/ou universo na qual um produto está inserido.

Sombra: semelhante à ferramenta anterior, porém aqui não há simulação. A equipe acompanha durante um período de tempo o usuário em suas atividades, monitorando e registrando pontos importantes para imersão.

3.2.2.3 Análise e síntese

Cartões de Insight: Semelhante ao sistema de post-it proposto por Brown. Nesta ferramenta são registradas reflexões e "insights" embasados nas pesquisas exploratórias realizadas na fase anterior. Pode-se entender essa ferramenta como a organização visual e resumo da fase de Imersão onde a seguir estes cartões são aplicados em um diagrama de afinidades. Os cartões ajudam a visualizar o todo, a inspirar e gerar novas ideias. Segundo Brown [5] o post-it, "incorpora a transição da fase divergente, que constitui a fonte da nossa inspiração, à fase convergente, que representa o mapa para nossas soluções".

Diagrama de Afinidades: organização dos cartões produzidos na atividade anterior, com base em afinidade, similaridade ou proximidade, dentro de um diagrama que contenha as macro e micro áreas abordadas no projeto. Normalmente utilizada quando se tem uma quantidade muito grande informação.

Mapa Conceitual: representação gráfica de todo projeto até o momento (pode/deve continuar por todo o projeto). Nele são registrados informações e insights produzidos e como interagem entre si, permitindo associação de dados e significados e extração de novas ideias, ou até mesmo visualizar possíveis erros e riscos. O mapa tem diferentes áreas como expectativas do usuário, jornada de atividade, mapa mental, erros e como contorna-los. A importância desta ferramenta como mostra Knudtson [30] é visualizar que alguns pré-conceitos do designer, cliente ou usuário não se adequam a

realidade, sendo possível também achar alternativas e insights para problemas futuros.

Critérios Norteadores: semelhante aos requisitos principais de um projeto de desenvolvimento de software. Pode ser considerado um resumo do mapa conceitual, com pontos chaves norteadores que não devem ser desobedecidos.

Personas: criação de personagens fictícios baseados nos arquétipos levantados na fase de imersão acerca dos usuários. Cada persona representa motivação, desejos, expectativas e necessidades de um determinado grupo de usuários que possuam semelhanças entre si. A partir de um questionário, caderno de sensibilização, etc é possível traçar um ou mais perfis do tipo de usuário que utilizará um produto/serviço, entretanto o uso da persona potencializa a aproximação e empatia da equipe ajudando a visualizar de forma mais concreta para quem se está desenvolvendo.

Mapa de Empatia: Semelhante ao mapa conceitual, a diferença é que nesta ferramenta é mapeado somente as motivações e expectativas do usuário, para proporcionar melhor entendimento do público-alvo. Pode ser considerado um sub-mapa integrante do mapa conceitual.

Jornada do Usuário: como nome diz, é a representação visual do caminho que o usuário percorre junto a um produto ou serviço. Esta ferramenta permite entendimento fracionado e detalhado de todas as atividades que o usuário irá realizar, conectado com suas expectativas e motivações. Aqui são utilizadas todas as informações recolhidas até o momento para a construção de uma jornada condizente com a realidade, sendo possível prever possíveis erros e obstáculos que o usuário terá de enfrentar.

3.2.2.4 Ideação

Brainstorming: uma ferramenta que pode potencializar a criatividade, o brainstorming é utilizado como um estímulo do pensamento visual. Brown [5] declara que as sessões de brainstorming são realizadas em salas exclusivas, onde as regras estão escritas na parede, sendo elas: Adie as críticas. Incentive ideias malucas. Mantenha-se concentrado no tópico. Tome por base as ideias dos outros [5]. Aqui a técnica de post-it pode ser empregada novamente.

Workshop de Cocriação: encontro organizado que reúne todos os possíveis atores de um projeto (equipe de criação, desenvolvimento, cliente e usuário) com uma série de atividades que visam criação colaborativa. Recomendada em momentos de impasse,

onde o maior número de ideias e visões se torna necessária para tomada de decisões.

Cardápio de Ideias: criação de um catálogo ou caderno de registro com todas as informações chaves produzidas/definidas até momento.

Matriz de Posicionamento: ferramenta para certificar-se de que decisões técnicas estejam alinhadas ao posicionamento estratégico do projeto, assim como com as personas. Organizam-se em uma matriz os pontos norteadores do projeto e verifica-se o alinhamento das decisões tomadas. É uma fase de validação das ideias que serão implementadas.

3.2.2.5 Prototipação

Uma das ferramentas essenciais do design thinking e de qualquer organização criativa. Para ele, o ato de se dispor a construir um objeto a partir de uma ideia é a melhor evidência da experimentação. O ato de construir protótipos faz com que resultados sejam obtidos com maior rapidez, contribuindo para convergir dentre as diversas possibilidades existentes.

Protótipos de Papel: representação de interfaces gráficas com diferentes níveis de abordagem. Como o nome diz esta etapa deve ser feita em papel, onde mudanças podem ser facilmente aplicadas e as soluções são representadas de forma bastante simplificada, o que permite a visualização do todo e suas interações.

Modelo de Volume: representação do produto mais detalhada, podendo até ter a aparência final. Uso de diversas camadas que possa ilustrar a interatividade das ferramentas.

Encenação: teste primário encenado que se utiliza do protótipo para testar uma ou mais atividades. Realizado pela própria equipe, podendo contar com usuário ou cliente, onde se tem um primeiro contato com o produto e suas funções primárias. Essencial para troca de feedback e previsão de erros futuros.

Storyboard: representação visual de uma atividade em forma de história em quadrinhos, para comunicar uma ideia complicada a terceiros ajudando-os a visualizar o encadeamento de uma solução. Bastante recomendado quando cliente não consegue entender o porquê de determinada decisão.

3.2.2.6 Implementação

A implementação é o terceiro espaço da inovação, onde a ideia deve ser transmitida a toda organização de forma clara o suficiente para ser aceita, "comprovando-a e mostrando que ela funcionará em seu mercado alvo" [5]. Nesse estágio, a prototipagem ainda é essencialmente importante, desta vez, a mesma se apresenta de forma mais completa, e complexa.

3.2.3 Inovação Incremental

De acordo com o dicionário Aurélio da língua portuguesa, inovar significa: introduzir novidades; renovar, inventar, criar. Já a palavra incremento se refere ao aumento, desenvolvimento. Portanto quando combinadas as duas palavras significam "alguma melhoria em algo", e é sobre isso que trata a inovação incremental. Com essa definição já é possível traçar um paralelo deste tipo de inovação com metodologias ágeis, que prezam por desenvolvimento compartimentado e incremental, porém neste caso a inovação é o centro do processo.

Para traduzir em termos de mercado inovação incremental de acordo com o Dicionário de Negócios é uma série de pequenas mudanças (melhorias) que ajudam a manter ou melhorar a competitividade de um produto, serviço ou processo. Normalmente utilizada por empresas de alta tecnologia que necessita melhorar seus produtos continuamente para atender as demandas dos usuários, deste modo as empresas querem inovar tendem a fazê-lo apenas um pouco de cada vez. Pense em inovação incremental como corte ou funcionalidade melhorias de custo em produtos ou serviços existentes [33].

A razão de a inovação incremental ser tão popular é porque ela tem um risco reduzido em comparação com a inovação radical. Como afirma Tidd, Bessant e Pavit [63] uma das dimensões da inovação é o grau de novidade envolvida, ou seja, há diferentes graus de novidade desde melhorias incrementais (contínuas) menores até mudanças radicais que transformam a forma como vemos ou usamos as coisas. Os estudos acerca do desenvolvimento do processo incremental sugerem que os ganhos cumulativos de eficiência são muito maiores em longo prazo do que aqueles obtidos com as mudanças radicais ocasionais. A inovação incremental é importante porque permite a empresas acompanhar a concorrência e manter suas posições de mercado.

De acordo com Incremental [27] do ponto de vista gerencial este tipo de

inovação deveria ser encorajado, pois fazer um produto 2% melhor ou mais barato ou mais rápido permite as empresas manter seus clientes e a continuar crescendo. Caso contrário a concorrência conquistará o público com suas melhores ofertas.

3.3 Interação Humano-Computador e Usabilidade

O campo de interação humano-computador (IHC), sendo uma atividade cada vez mais ligada ao design [72] encontra problemas similares aos citados até então, portanto ofertar um framework de processos IHC que esquematize as relações internas visando aumentar níveis de interação, reflexão, entendimento mútuo, criatividade e inovação. Ebenreuter [15]aponta que como resultado isto pode prover aos profissionais de TI e Design grande insight a respeito da disciplina de IHC assim como uma visão mais abrangente de todos os conhecimentos envolvidos no desenvolvimento de software.

A interação com computadores e outras tecnologias tornou-se habitual e indispensável devido ao crescente uso das mesmas nas diferentes esferas do cotidiano. Esta interação não ocorre de maneira direta, ou seja, entre o usuário e o equipamento sempre haverá um intermediador, que passa a se chamar de interface, que começou como controles analógicos e hoje foram substituídos por interfaces digitais.

Neste sentido, Barros [2] afirma que a necessidade dos usuário em acessar novos sistema com rapidez e eficiência, fez com que os estudos dos processos que envolvem a interação homem-máquina recebessem grande importância. Sendo que nesta relação homem-máquina, o usuário (homem) deve ser o centro (objetivo) das atenções, e deve ser tão estudado quanto as tecnologia emergentes.

De interação humano-computador se é levado a discussão mais específica de interface humano-computador, que segundo Carvalho [6] "a interface homem-computador se refere a interface que serve de interconexão entre dois sistemas que trocam informações, sendo eles: de um lado o computador e de outro o ser humano."

No desenvolvimento de uma interface deve-se levar em conta os conceitos aqui abordados, de design universal, usabilidade, acessibilidade e IHC. Para obter uma interface harmoniosa, Cybis [11] apresenta uma lista de atributos desejáveis em uma interface, baseado em literatura especializada. Os atributos são:

- *Diversidade:* a interface deve suportar os diferentes tipos de usuários e se adaptar a estas necessidades específicas;

- *Complacência:* a interface deve permitir que usuário desfaça ações acidentais e considerar esquecimento de informações;
- *Eficiência:* minimizar esforço realizado para executar tarefas;
- *Conveniência:* fácil acesso a todas as operações importantes;
- *Flexibilidade:* prover maneiras diferentes de se executar a mesma ação;
- *Consistência:* comportamento e apresentação física devem ser guiados por regras definidas e de senso comum;
- *Prestimosidade:* a interface deve ser prestativa, fornecer ajuda quando requisitada e perceber quando usuário estiver com dificuldade;
- *Imitação:* a interface deve imitar o mundo real e os diálogos humanos;
- *Naturalidade:* comunicação de maneira natural, sem exigir terminologias especificas à uma área;
- *Satisfação:* satisfazer o usuário, sem demora ou frustação;
- *Passividade:* a interface deve assumir papel passivo, permitindo que usuário detenha o controle da interação.

O objetivo do IHC, segundo Ogawa [48] é elaborar métodos de análise das relações homem-computador, e elencar resultados para que possam ser acessados por desenvolvedores e projetistas, de forma a verificar se o sistema criado (ou vir a ser) é compatível com a necessidade dos usuários.

A Interação com novas tecnologias e todos os seus sistemas tornou-se rotineiro e imprescindível para a manutenção da sociedade. Quando esta interação exige dos usuários capacidades além dos seus limites, pode ocorrer a exclusão destes indivíduos ou a extinção de determinado sistema que não se adapta as necessidades e habilidades compartilhadas pela maioria. Portanto, mais do que uma estratégia de marketing para aumentar vendas e conquistar público, a usabilidade como aponta Nielsen e [45] "serve para fortalecer a humanidade e tornar mais fácil e mais agradável à tecnologia que impregna cada aspecto da vida moderna".

A usabilidade de acordo com Cybis [11] são as características que se atribuem ao uso de um sistema interativo, que diz respeito ao usuário, a interface, o equipamento, a tarefa e as demais interferências do ambiente (contexto) ao qual fazem parte. Deste modo, na avaliação de um sistema levam-se em conta diversos fatores para averiguar a eficiência, eficácia, consistência e satisfação obtida com o mesmo.

Em relação aos atributos que fazem de um software ser bem-sucedido Dias [14]

indica que a eficácia é a principal motivação que levará os usuários a utilizarem o software de novo, uma vez que mesmo possuindo uma interface bem resolvida e de fácil uso, se o programa não atingir os objetivos necessários ele estará destinado ao fracasso.

Para se desenvolver um sistema é necessário um estudo aprofundado das relações que ocorrem entre o usuário e o software, assim como prever todo tipo de interferência que estes podem vir a sofrer do contexto em que estão inseridos. A realização de tal estudo ocorre por meio de pesquisas de campo e consultas em literaturas especializadas que evidenciam parâmetros e princípios que norteiam o desenvolvimento de interfaces.

No que diz respeito ao projeto de interfaces Shneiderman et al. [59]aponta oito "regras de ouro", estas são:

- *Consistência:* Sequência de ações similares para procedimentos similares. Manter um padrão visual para as cores, layout e fontes. Utilizar a mesma terminologia em menus;
- *Atalhos para usuários assíduos:* Teclas de atalho, macros e navegação simples facilitam e agilizam a interação do usuário mais experientes com a interface;
- *Feedback informativo:* Toda e qualquer ação do usuário requer uma resposta do sistema, cujo qual será mais ou menos explicativa dependendo do tipo de ação a ser executada;
- *Diálogos que indiquem término da ação:* As sequências de ações do sistema devem ser organizadas de tal forma que o usuário consiga entender os passos e saiba quando cada um deles for executado com sucesso;
- *Prevenção e tratamento de erros:* A interface não pode dar vias para o usuário cometer erros graves, e caso ocorram erros, deve haver mecanismos que tratem, corrijam na medida do possível, e caso não seja possível, instruam o usuário para uma possível solução;
- *Reversão de ações:* Sempre que possível, as ações devem ser reversíveis, de forma que tranquilize o usuário e lhe dá mais coragem para explorar o sistema;
- *Controle:* Os usuários mais experientes devem ter a sensação de que eles dominam os processos do sistema, que apenas responde à suas ações;
- *Baixa carga de memorização:* O sistema deve conter uma interface simples para memorização. Para isso requer uma boa *estrutura* e *equilíbrio* para relacionar elementos e facilitar a memorização subjetiva das telas, sem exigir esforço.

O método de avaliação heurística (tabela 3) conta com dez recomendações

condensadas por Nielsen [44] que são:

Tabela 3: Heurísticas de Nielsen [44]

Heurística	**Descrição**
Visibilidade de Status	O sistema precisa manter os usuários informados sobre o que está acontecendo, fornecendo feedback dentro de um tempo razoável.
Compatibilidade com o Mundo Real	O sistema precisa falar a linguagem do usuário, com palavras, frases e conceitos familiares. Seguir convenções do mundo real, fazendo com que a informação apareça numa ordem natural e lógica.
Controle do Usuário e Liberdade	Usuários frequentemente escolhem por engano funções do sistema e precisam ter claras saídas de emergência para escapar do estado indesejado, sem ter que percorrer um extenso diálogo. Prover funções desfazer e refazer ação.
Consistência e Padrões	Diferentes palavras, situações ou ações não podem significar a mesma coisa. Seguir convenções de plataforma computacional.
Prevenção de Erros	Melhor que uma boa mensagem de erro é um design cuidadoso na qual previne o erro antes dele acontecer.
Reconhecimento ao invés de lembrança	Tornar objetos, ações e opções visíveis. O usuário não deve ter que lembrar a informação. Instruções para uso do sistema devem estar visíveis e facilmente recuperáveis quando necessário.
Estética e Design Minimalista	Diálogos não devem conter informações irrelevantes ou raramente necessárias.
Suporte aos Usuários no Diagnóstico e Correção de Erros	Mensagens de erro devem ser expressas de forma clara, indicando precisamente o problema e possíveis soluções.
Ajuda e Documentação	Deixar sempre disponível a documentação de ajuda para o usuário. Essas informações devem ser fáceis de encontrar, focalizadas na tarefa do usuário e não muito extensas.

3.4 Design Thinking e Desenvolvimento Ágil

Aqui será levantado o uso das teorias de design dentro da área de TI. No que diz respeito a convergência e uso efetivo do design thinking será resumido a seguir os resultados encontrados.

No estudo realizado por Bartolomeu [3] da aplicação do DT em uma empresa de TI o autor aponta nos resultados, que o ambiente da empresa se tornou muito mais

aberto à mudanças e pró criatividade/inovação. Além disso os projetos tornaram-se mais centrados no usuário e suas necessidades, aumentando satisfação e níveis de venda. Resultados similares foram encontrados no estudo conduzido por Chasanidou, Gasparini e Lee [9], como na experiência controlada em workshops de desenvolvimento que enfatiza o aumento na troca de *insights* proporcionada pelo *design thinking*.

Dentre as principais ferramentas e aplicações de design thinking citadas na literatura temos o modelo clássico de 5 passos do DT [3]: empatizar, definir, ideação, prototipar e testar: Nota-se também o extensivo uso de personas e mapas de clientes para direcionar o projeto de desenvolvimento, assim como mapeamento de jornada do usuário pela equipe, demarcando pontos críticos envolvendo cada área de criação e programação [69]. Um dos estudos [49] é destacado por desenvolver toda uma metodologia pautada no *storytelling* que além de potencializar a experiência do usuário, o sistema proporciona insights sobre o processo conforme este se desdobra.

Em um estudo realizado na Inglaterra [16] o autor relata o ensino do design thinking para profissionais de computação também pautado no modelo de 5 passos do processo de DT, relutantes no início os estudantes compartilharam suas impressões a respeito do uso desta metodologia e novamente apontam que o uso do design thinking potencializa a comunicação em equipe, e ajuda a abordar o problema de forma abrangente, ao invés de se pautar em preconcepções individuais. Ressalta-se que neste e em outros há menção ao fato do design thinking ser o contrapeso das metodologias um tanto enfadonhas da gestão de TI, fomentando um ambiente mais criativo, quebrando a monotonia de modelos de desenvolvimento como a cascata (waterfall) ou integração contínua [22].

Em artigos como [10] a integração da gestão de design aos métodos da tecnologia da informação, é proposta por meio de um modelo mais puro de gestão, ou seja vê-se aplicada à empresas a gestão de design organizacional pura. Não há menção de como as metodologias internas de produção e processo são afetados (se é que são). São dois casos interessantes ao abordarem o design na área da TI desde os níveis estratégicos, porém não sugerem a convergência de disciplinas.

Já em outros estudos como [22] verifica-se a fusão de duas metodologias, entretanto o que se relata é apenas a integração de pequenas ferramentas de uma no grande corpo metodológico da outra, como um forma de melhorar resultados, seja em qualidade, fomentar ambiente criativo, produção lúdica [20] ou aumentar a produtividade, ou seja uma disciplina sempre rege e lidera enquanto outras servem

como argamassa para preencher lacunas.

Apenas um dos estudos [69] utiliza a ideia de convergência de teorias, neste caso três disciplinas (Design Thinking, Agile e Lean Startup) são modeladas e organizadas em uma metodologia para gerenciar projetos de software. Neste caso cada disciplina mantêm sua independência, e não há dominação de uma área sobre a outra. As três são coordenadas ao longo do projeto sendo identificadas o momento que cada um deve atuar. Os resultados do estudo sugerem que mantendo a integridade de cada disciplina é possível combinar as metodologias e abordá-las em rodadas, permitindo altos índices de inovação e bom relacionamento com usuário.

Similarmente alguns estudos aplicam à dinâmica de trabalho a metodologia de *gamificação* (Seção 3.5), uma das novas vertentes do design emocional. A proposta da gamificação é tornar processos monótonos de desenvolvimento mais atrativos, fazendo a manutenção dos níveis de interesse dos profissionais por sistemas de recompensa a cada nível de sucesso alcançado. Os resultados apontam que ao adicionar o elemento "diversão" na rotina de trabalho permite aos profissionais enxergarem claramente o progresso individual e coletivo, uma vez que ambientes gamificados geram logs (relatórios) sobre o processo em si, de muit*a valia para a equipe.*

3.5 Gamificação como Análise da Motivação

Milhares de pessoas se ocupam diariamente em resolver tarefas inventadas à procura de realização e o sentimento de conquista proporcionado por jogos eletrônicos. Este poder de engajamento certamente não passa despercebido e atualmente há grandes discussões sobre a aplicação de elementos utilizados em *games* como ferramentas para impulsionar rendimento e produtividade, sejam na educação, marketing ou ambientes organizacionais. Deterding [13] aponta que seguindo estas observações especialistas passaram a ver jogos como uma nova fonte de ferramentas heurísticas para desenvolver interfaces e metodologias de trabalho mais imersivas e com aspecto lúdico.

Uma definição bastante abrangente de gamificação é o uso de elementos de game design em contextos alheios a este. Nicholson [43] mostra que o uso recorrente desta metodologia se dá com a integração de elementos de pontuação presente em jogos, como pontos, níveis, objetivos gerais e específicos em ambientes organizacionais. Os benefícios desta metodologia estão na mudança comportamental das pessoas envolvidas, direcionando e otimizando a força criadora aos objetivos propostos de modo a alcançar as recompensas definidas.

O processo de gamificação pode parecer à primeira vista puro potencial positivo, que só viria a aperfeiçoar um ambiente de trabalho. Porém como pode ser verificado em Deterding [13] e Zichermann e Cunningham [71] quando aplicado de forma superficial o efeito desta metodologia é exatamente o inverso. O ponto central da gamificação é MOTIVAÇÃO, portanto se um sistema somente determina os objetivos, fornece pontuações e recompensas externas ao processo em questão, o que se nota é a gradativa queda de motivação interna dos indivíduos, que acabam se tornando "dependentes" de uma constante gratificação que por si afeta a média de produtividade dos mesmos.

Nestes casos Deterding [13] afirma que o problema está exatamente em sistemas que somente aplicam elementos de *games* dentro um framework de trabalho, ao invés de criar este framework pautado no *game design,* ou seja, jogos não são uma alternativa a uma experiência imersiva, mas sim uma *lente sob a qual o processo pode ser analisado.*

Uma das maneiras de tornar a experiência de gamificação mais significativa é permitir que os indivíduos envolvidos estipulassem seus próprios objetivos. Nicholson [43] aponta que neste caso o desafio em estabelecer este framework gamificado está exatamente em definir parâmetros que guiarão estes indivíduos a propor objetivos de curto e longo prazo alinhados as metas universais da atividade em questão.

Rose e Meyes [53] aponta uma solução para criar este sistema-guia. Primeiramente apresentar de diversas formas ou meio o que deve ser compreendido. Segundo apresentar os possíveis caminhos que podem ser percorridos para alcançar o objetivo. E por último internalizar os resultados para que se entenda o porquê do processo que se acabou de executar. Portanto, garantindo uma variedade de caminhos para o estabelecimento do onde, como e o porquê, permite aos indivíduos criarem experiências mais significativas dentro um projeto, distribuindo o poder de decisão e escolha.

3.5.1 Motivação

Jogos são ótimas fontes de motivação, Zichermann e Cunningham [71] afirmam que ao focar em três elementos centrais, *prazer*, *recompensa* e *tempo*, jogos se tornaram uma força poderosa para engajar indivíduos em atividades que antes não os interessava.

Para a gamificação ser significativa, ou seja, que proporciona níveis consideráveis de motivação Nicholson [43] aponta que é importante levar em

consideração o background que o usuário traz à atividade e o contexto organizacional no qual a atividade está situada. Um desafio grande na criação deste tipo de sistema é o desenvolvimento de uma estratégia para abranger uma grande variedade de *backgrounds*, desejos e habilidades do usuário.

Quando se trata de motivação em alguma atividade é comum se deparar com o conceito de *flow*, que é alcançar um estado de concentração grande, causado por uma motivação que flutua entre excitação e tédio. Na figura 9 podemos ver que o fluxo ocorre quando o grau de desafio de uma atividade é proporcional a habilidade do indivíduo, gerando motivação e o estado de intensa concentração e produtividade.

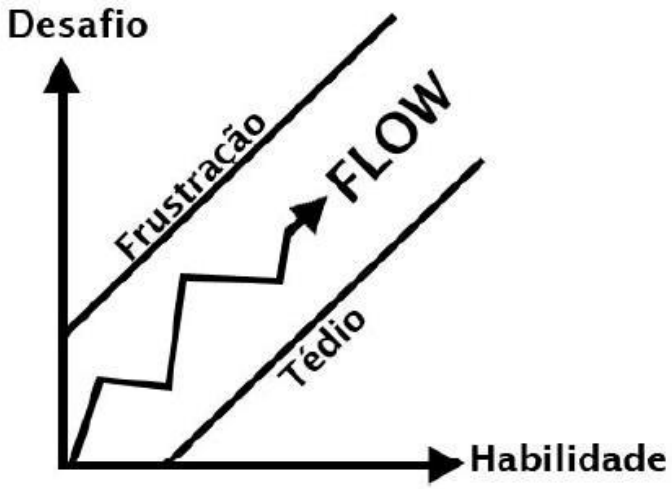

Figura 9: Zona de fluxo [64]

Zichermann e Cunningham [71] alertam para o fato de que muitas empresas falham ao empregar a gamificação, pois acham que podem substituir recompensas reais e incentivos palpáveis com pontos e troféus vazios em valor ou significado, e manipular as pessoas a executarem atividades que elas não querem fazer. Portanto não adianta gamificar uma atividade tediosa e laboriosa esperando torná-la agradável e divertida, todo o processo que irá ser gamificado deve anteriormente ser analisado e reformulado para corrigir problemas críticos e definir quais as recompensas e incentivos reais que podem ser já implementados mesmo sem o uso da gamificação.

Deterding [13] propõe que para cada implementação de gamificação deve-se primeiramente realizar um levantamento completo dos objetivos e processos de uma organização (ou tarefa), quais são as atividades realizadas que agregam e geram valores para o negócio e por último quem são os indivíduos e quais são suas motivações internas ao se envolverem com determinado negócio.

Retomando o conceito de gamificação, no livro Gamification by Design de Zichermann e Cunningham [71] são apresentados quatro tipos principais de jogadores (traços de personalidade), que se remete ao que motiva as pessoas a jogarem, e estes são:

- Exploradores (50%): de forma geral são jogadores que gostam de explorar mundos e descobrir coisas novas para mostrar a comunidade. Para eles a experiência é o objetivo;
- Conquistador (40%): são jogadores que gostam de vencer e adoram competição. O problema em desenvolver um sistema exclusivo para este tipo de personalidade é que nem sempre se pode ganhar e conquistar e para conquistadores perder pode causar desinteresse e frustração;
- Sociais (80%): a maioria dos jogadores pode ser classificada como sociais. É o tipo de jogador que jogam para poderem interagir socialmente com outros jogadores. Não significa que estes não se interessem em vencer, porém o foco principal está na interação e conexão entre jogadores que um sistema pode proporcionar;
- Matadores (20%): são o menor grupo de jogadores dentre as quatro principais personalidades. Para este grupo, assim como conquistadores, ganhar é imprescindível, entretanto isso não basta alguém tem que perder e para completar várias pessoas precisam assistir esta vitória e expressar admiração para satisfazer este tipo de jogador.

Vale ressaltar que individualmente as pessoas transitam entre estes traços de personalidade, ou seja, até mesmo quem é prevalentemente matador ainda procura jogos para satisfazer necessidades sociais ou para objetivos de exploração. Todas os traços mencionados até então não devem ser interpretados de forma polarizada, há nuances entre elas e o que autor relata é que o sistema gamificado neste caso tem importante papel ao fomentar (trazer a tona) uma ou mais das motivações relatadas.

4 PROCESSOS METODOLÓGICOS

Pesquisa de natureza aplicada, uma vez que será gerado no final um mapeamento completo sobre o problema que resultará no desenvolvimento do aplicativo/plataforma interativa, possui caráter exploratório, que segundo Cervo, Bervian e Silva [8] estabelecem critérios, métodos e técnicas para a elaboração de uma pesquisa e visa oferecer informações sobre o objeto desta e orientar a formulação de hipóteses. Envolve (a) levantamento bibliográfico, já realizado; (b) análise de similares; e (c) levantamento e análise de dados que estimulem a compreensão.

Finalizado o levantamento bibliográfico acerca do problema e possíveis soluções, dar-se-á início aos procedimentos diretamente relacionados ao projeto, começando com levantamento de dados anteriores à produção (análise de similares), desenvolvimento do framework e ferramentas de Design, testes e readequação da proposta.

Das etapas tem-se:

- Levantamento bibliográfico das metodologias Agile, Design Thinking, Motivação e suas ramificações;
- Levantamento de dados: análise de similar no mercado, buscar documentações existentes de grandes empresas;
- Estudo de caso com projetos realizados no laboratório de software GAIA e sua dinâmica de trabalho;
- Levantamento de dados: questionários com profissionais, acerca dos desafios de concatenar design e programação;
- Criar proposta de framework e suas respectivas ferramentas para auxiliar o desenvolvimento de software;
- Fase de testes e análise dos resultados obtidos para implementação final do framework no laboratório GAIA soluções em TI.

4.1 Análise de Similar: Google Material

Quando se fala em Google é difícil não se lembrar de mudanças. A empresa investe fortemente quando o assunto é inovação, sempre procurando nichos a serem explorados e constantemente atualizando seus processos. Apesar de ser caracterizada por quebrar paradigmas de mercado e inserir soluções radicais de produtos e serviços,

ao se observar mais de perto é possível verificar que a empresa foca suas energias em melhorias contínuas, tanto que em uma matéria publicada com seus oito *pilares de inovação,* é possível notar o interesse da companhia em sempre começar “pequeno” e expandir os negócios. Os pilares são:

- Pense grande, mas comece pequeno;
- Empenhe-se por inovação contínua, não perfeição instantânea;
- Procure por ideias em todos os lugares;
- Compartilhe tudo;
- Crie com imaginação, concretize com dados;
- Seja uma plataforma;
- Nunca deixe de falhar.

Como exemplo, eles citaram o Google Books, que começou com uma ideia pequena e apenas uma pessoa e um scanner, mas ao ser identificado o potencial da ferramenta, a mesma começou a evoluir lentamente e hoje conta com mais de dez milhões de livros disponíveis online.

Outro exemplo de inovação incremental, podemos citar o lançamento do Google Gmail, considerado por muitos como o melhor serviço de e-mail do mercado, é um exemplo de dedicação à inovação incremental. Quando o Gmail foi lançado tinha um conjunto limitado de funcionalidades, mas fez uma coisa muito bem, entregava e-mails. Ao contrário dos concorrentes, era limpo e fácil de usar, sem anúncios em flash que distraem e inúmeras melhorias de interface.

Ao longo do tempo o Google liberou mais funcionalidades e tornou o serviço melhor, mais rápido e fácil de usar. Sempre levando em conta as necessidades do usuário. Há pouco tempo ele deixou de ser "beta" e finalmente foi listado como um serviço completo, apesar de até hoje estar sendo melhorado continuamente.

A empresa tem usado esta estratégia para muitos de seus produtos, como por exemplo, o Google Maps e o navegador Google Chrome. Essa dedicação a melhoria contínua mantêm os produtos Google:

- Relevantes para o consumidor;
- Extremamente competitivos no mercado;
- Focado na redução de custos.

A principal vantagem da inovação incremental se faz presente aqui, e é fato de

que essa estratégia dificilmente dará errado. Seja reduzindo custos ou melhorando o produto nada acontece de forma dramática que o público possa rejeitar todo o conjunto. No pior dos cenários os usuários simplesmente não irão responder ao produto e no melhor caso haverá um enorme crescimento em vendas, popularidade e valor de marca.

Grande diferencial da Google é a ampla divulgação de robusta documentação que serve como guia de produção para desenvolvedores de software. O Google material surgiu como uma tentativa da empresa de fomentar desenvolvimento de softwares mais unificados dentro de suas plataformas. Pode ser considerada uma doutrina e oferece um conjunto imenso de regras, dicas e ferramentas de apoio para produção de interfaces. Com linguagem híbrida voltada tanto para designer como programadores.

Os objetivos da doutrina material segundo a própria empresa são:

- Criar linguagens visuais que sintetizem princípios clássicos de design com inovação e aplicação de novas tecnologias;
- Desenvolver um sistema de métricas único que unifique a experiência do usuário nas diferentes plataformas e tamanhos de dispositivos.

Na documentação é possível encontrar métricas visuais de design que falam tanto para designer como programador (figura 10), assim como códigos prontos, auxílio para adição de módulos, ou seja, auxilia tanto no entendimento estético do programado quanto técnico para o designer.

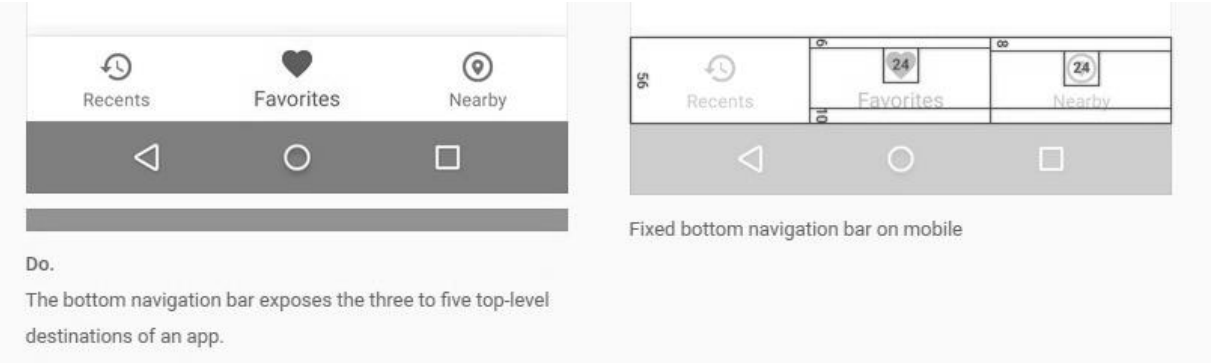

Figura 10: Google material [37]

4.2 Análise do Framework GAIA

O framework atual do GAIA como esperado possui abordagem totalmente voltada à área da TI. A análise a seguir serve para levantar pontos críticos no framework no que tange a falhas entre processo e realidade do laboratório (*Engenharia de Software*), pontos críticos de ausência de Design (*Gestão de Design*) e fatores que geram

desmotivação e falta de engajamento (*Gameficação).* Nas etapas seguintes será desenvolvida a convergências das ferramentas de Design Thinking no framework GAIA Agile.

O padrão atual do framework de trabalho (figura 11) pode ser definido como modelo linear cascata de produção, porém pelo contexto de laboratório, as etapas e tarefas definidas apresentam flexibilidade quanto à quando devem/podem ser utilizadas, mesmo que de forma aleatória e tendo como parâmetro somente a decisão do gerente.

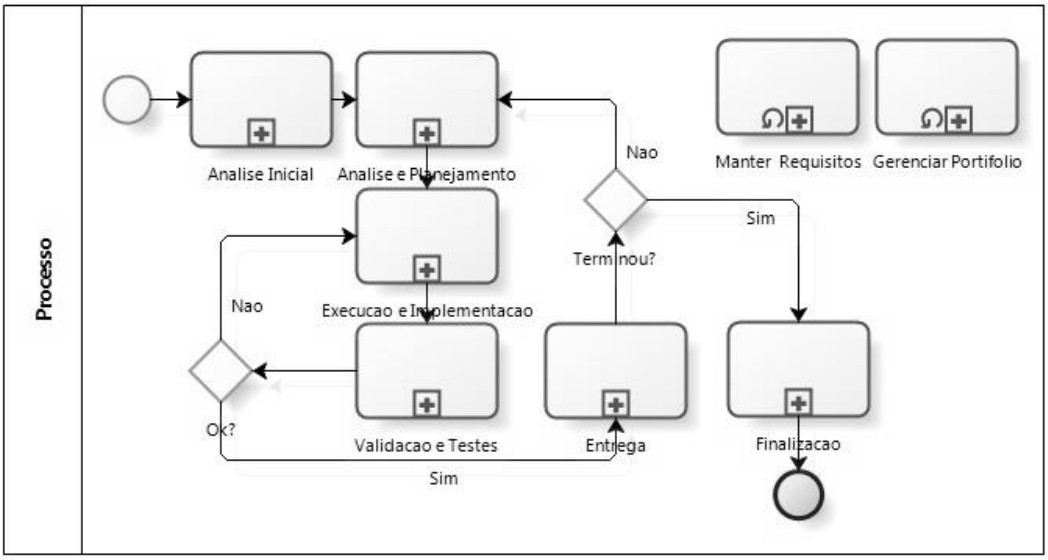

Figura 11: GAIA PDS [21]

Como visto no capítulo sobre desenvolvimento ágil, equipes pequenas, multidisciplinares e principalmente em um ambiente de fomento a construção de conhecimento como o laboratório GAIA, a adoção de um framework já estruturado em iterações flexíveis e que tem como base principal a **comunicação** entre os membros da equipe, se torna praticamente indispensável.

No modelo atual uma etapa ou atividade normalmente só começa quando a anterior foi fechada e validada, criando o efeito cascata que como foi visto, em grupos pequenos e com menos experiência acarreta ao acúmulo de erros muito grandes que só são percebidos no final. De modo a enquadrar o processo em um modelo ágil, primeiramente será feita análise de cada atividade e cada fase para detectar falhas, a seguir será realizada levantamento de projetos reais com depoimentos de profissionais para validar esta análise e por fim o framework será redesenhado nos padrões Agile.

Análise Inicial (tabela 4): totalmente em controle do gerente. O desenvolvedor

só tem acesso às informações do projeto na última etapa, a equipe não tem poder de decisão nos estágios iniciais. Este tipo de processo seria condizente se tratando de uma equipe fixa onde o gerente tem total conhecimento dos *assets* que possui. No caso da GAIA, equipes temporárias se tornam desafio por este motivo e a participação de membros de desenvolvedores em fases iniciais é necessária.

Nesta etapa é essencial a aclimatização da equipe, onde uma dinâmica de grupo colaboraria com a troca inicial de ideias e exposição das habilidades técnicas de cada um, o que permitiria ao gerente um insight maior sobre a equipe que tem em mãos e como as etapas seguintes serão influenciadas por conta desses fatores humanos.

Tabela 4: Análise inicial

Etapa	Participantes
Reuniões	Cliente, Analista de Sistema, Gerente de Projeto
Estabelecer Escopo	Cliente, Analista de Sistema, Gerente de Projeto
Estimativas	Gerente de Projeto
Preparar Ambiente	Gerente
Analisar Viabilidade	Gerente
Criar Project Charter	Gerente
Reunião Kick-off	Gerente, Desenvolvedor, Suporte, Cliente.

Análise e Planejamento (tabela 5): ainda permeia aqui a completa ausência de desenvolvedores que não têm poder de decisão em relação à forma como irá trabalhar. Como demarcado na seção de Gameficação, um ponto crítico na geração de motivação é permitir aos indivíduos construir sua jornada de trabalho, dar um mirante e ferramentas de suporte, porém permitir a cada um expor seus métodos e valores pessoais.

Em equipes temporárias tal fator se torna problemático, pois o gerente não tem domínio/conhecimento suficiente sobre os membros da equipe. Outro ponto é que a falta de delegação de poder de decisão acarreta na **motivação** e **engajamento** dos desenvolvedores, além da inabilidade dos mesmos de enxergar ou mesmo definir os requisitos apropriados ao projeto.

Tabela 5: Análise e planejamento

Etapa	Participantes
Levantar Requisitos	Analista
Expandir WBS	Gerente
Revisar Requisitos	Analista
Validar Requisitos	Analista e Cliente
Definir Entregas	Gerente e Analista
Manter Rastreabilidade dos Requisitos	Analista
Planejar Indicadores	Gerente
Recursos	Gerente
Análise de Riscos	Gerente
Estimar Prazos e Custos	Gerente
Elaborar Cronograma	Gerente
Elaborar Tarefas	Gerente
Planejar Testes	Gerente
Planejar Configuração	Gerente
Revisar Planos	Gerente
Disponibilizar Informações	Gerente
Definir Informações	Gerente
Definir Fase	Gerente
Analisar Viabilidade	Gerente

Execução e Implementação (tabela 6): participação assídua do desenvolvedor na etapa de produção. Enfatiza o caráter operacional do desenvolvedor, que não é incluído na tomada de decisão. Gera desmotivação e carga de trabalho grande, pois ocorre todo de uma vez e não de forma fragmentada. Além da ausência do designer, tem-se também a falta de atividades que fomentam criatividade e dinâmica de grupo mais flexível e aproximação dos membros da equipe.

O controle da comunicação é discutido tardiamente no projeto, sendo que deve ser tópico de discussão ao decorrer de todo o processo.

Tabela 6: Execução e implementação

Etapa	Participantes
Executar Tarefas	Gerente, Desenvolvedor, Analista
Gerenciar Riscos	Gerente
Garantir Qualidade	Gerente
Gerenciar Comunicação	Gerente, Desenvolvedor, Suporte, Cliente.
Gerenciar Configuração	Gerente

Validação e Testes (tabela 7): Testes só ocorrem na penúltima fase do projeto gerando acumulo de erros e retrabalho. Além disso, é teste unitário que não envolve

cliente/usuário que são ponto crítico para qualquer teste. Testes são fontes imprescindíveis de feedback e motivação para equipe, pois é somente ao atestar o potencial de um serviço/objeto é que a equipe realmente começa a acreditar no trabalho a ser desempenhado.

Tabela 7: Validação e testes

Etapa	Participantes
Executar Teste Unitário	Tester e Desenvolvedor
Análise de Resultados	Tester
Realizar Correção	Desenvolvedor

Entrega (tabela 8): Teste final e único novamente. Ocorre apenas no final do projeto, acarretando em carga grande trabalho de correção ou até mesmo falha do projeto caso algum erro grande não tenha sido detectado anteriormente.

Tabela 8: Entrega

Etapa	Participantes
Executar Teste de Integração	Tester e Gerente
Análise de Resultados	Tester
Implantar Resultado da Fase	Desenvolvedor
Reunião de Feedback	Gerente, Desenvolvedor, Suporte, Cliente

Vale ressaltar que como o autor deste trabalho é participante assíduo em projetos do GAIA, a metodologia aqui apresentada, como já foi dito, apresenta informalmente uma flexibilidade e tem caráter iterativo quando determinados requisitos não são atingidos. Porém a apresentação atual do framework enfatiza caráter linear de etapas enquadradas em caixinhas e o laboratório se beneficiaria de uma metodologia que ofereça desde começo trabalho adequado a realidade dos profissionais e estabelece de forma mais consistente um desenvolvimento iterativo e incremental que minimize erros e potencialize o aspecto de fomentação do conhecimento.

4.2.1 Análise de Projeto: App Câmbio Monetário

Projeto realizado ao longo do ano de 2017, em linhas gerais, o software permite a compra e manipulação de moedas de outros países, de forma a automatizar e facilitar este processo para os usuários. Para manter o anonimato, nenhum detalhe sobre a ferramenta ou sobre a empresa que solicitou a criação do software será fornecido neste texto. Será abordado apenas o processo de desenvolvimento de software empregado no

projeto e a opinião sobre o mesmo por um profissional de TI e Design.

O projeto contava com um gestor de TI, um analista de design, um programador e um designer de interfaces. Do processo tem-se:

- *Briefing* **A5***:* Reunião com o cliente para o levantamento de requisitos.
- *Reunião Kick-off* **A5***:* Reunião da equipe para discussão dos requisitos e detalhamento do projeto. Definição da plataforma que seria inicialmente trabalhada (Módulos Puros de Android), assim como aspectos técnicos e definição de fases e tarefas iniciais. O objetivo desta conferência foi a apresentação dos envolvidos e o levantamento de requisitos do projeto. De forma bem simplificada, foram apresentados os objetivos da ferramenta, fazendo um paralelo com ferramentas já existentes, bem como definições de prazos e o intervalo de tempo entre cada reunião.
- *Pesquisa Desk* **D1***:* Pesquisa de materiais relacionados aos requisitos, para definição da estética e funcionalidades do App.
- *Definição de Fases* **D2***:* Divisão do APP em quatro grandes áreas e fases de desenvolvimento, seguindo design > programação > design.
- *Definição de Tarefas* **D3***:* Estabelecimento do sistema de troca de arquivos e guia de produção entre designer e programador.
- *Design Conceitual (Protótipo)* **D4***:* Desenvolvimento do protótipo conceitual para discussão com cliente e programador. Definição da estética final e detalhes técnicos de produção (como exportação de gráficos, códigos de cores, grid de tela, linguagem correta para transferência de dados).
- *Análise pelo cliente* **C1:** Apresentação da proposta ao cliente seguido de feedback do mesmo.
- *Correções* **E1***:* Correção de telas (layout ou código) baseado no feedback do cliente. Sem testes reais com usuários.
- *Desenvolvimento* **P1***:* Desenvolvimento de todas as telas dentro das fases pré-determinadas, seguidas de análise do cliente. Este processo foi realizado em três etapas: (1) a definição do fluxo de interfaces e protótipos, (2) a codificação do software e (3) o teste, a validação e correções de erros da ferramenta.
- Governança (**A1 à A10**): atividades relacionadas a gestão de projeto, incluem: Reunião, Estabelecer Escopo, Estimativas, Preparar Ambiente, Analisar Viabilidade, Criar Project Charter, Reunião Kick-off, Expandir WBS, Validar

Requisitos, Definir Entregas, Rastreabilidade, Governança (que engloba Planejar Indicadores, Recursos, Análise de Riscos, Prazos e Custos, Revisar Planos, Análise Viabilidade), Gerenciar Qualidade, Gerenciar Comunicação.

A definição do fluxo de interfaces e protótipos foi uma tarefa realizada pelo designer. Para manter a consistência entre "o que foi idealizado" e "o que é possível desenvolver" (considerando os prazos e complexidade do projeto e de desenvolvimento), utilizou-se o *Material Design,* proposto pelo Google. Basicamente, esta especificação auxilia a criação das interfaces com base na usabilidade e experiência do usuário.

A codificação do software foi realizada por um profissional da área de TI com conhecimentos em Java e Android. Vale ressaltar a falta de experiência do profissional em relação ao Material Design. Isso fortalece o fato de que esta especificação é uma ótima ferramenta para a comunicação entre o designer e o programador. De forma geral, a arquitetura do software se resume ao uma Web Service em Java e um aplicativo desenvolvido em Android. O aplicativo faz requisições para o servidor, que às processam e retorna uma resposta.

De acordo com as tarefas acima, na figura 12 foi detalhado em um gráfico como ocorreu às iterações dentro do projeto, delimitando a participação do *designer*, *programador* e *gestor*. O que se tornou evidente no gráfico é o desalinhamento entre a produção do designer com a do programador dentro das fases pré-definidas. Enquanto o designer iniciava uma fase o programador ainda está executando ou realizando correções que também necessitava do designer. Os feedbacks do cliente também estavam fora de sincronia, uma vez analisado o layout conceitual este emitia uma série de correções que de fases já fechadas.

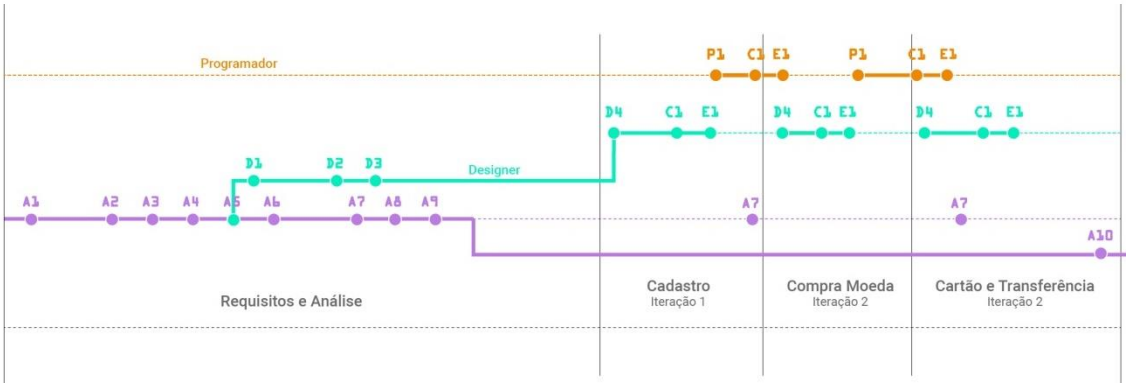

Figura 12: Processo app câmbio

Dos problemas citados pelos profissionais envolvidos, tem-se:

- De forma geral não houve testes no aplicativo no qual o designer pudesse comprovar suas decisões, prevalecendo somente "achismos" do cliente em relação às funcionalidades, quando testes poderiam ter sido de grande valia na comprovação destes dados e na definição mais consolidada de funções;
- O desenvolvimento não seguiu um padrão correto de produção passando por *Protótipo Conceitual de Layout > Desenvolvimento Primário de Funcionalidades (TI) > Testes e Correções > Avaliação do Cliente Pautada em Documentação Fundamentada das Fases Anteriores*. O que se sucedeu foi análise primária do cliente do protótipo conceitual que gerou problemas de interpretação pela impossibilidade do cliente de visualizar como os elementos gráficos e telas se relacionavam em tempo real (Um aplicativo funcional interativo é diferente de uma imagem estática, onde o exercício de imaginação tem que ser muito grande para se entender as diversas relações entre telas);
- Outro problema que ocorreu devido ao que foi citado anteriormente de o cliente ter contato com layout conceitual sem ter a mediação do programador e uma instância operante do APP com layout aplicado, é que a versão final não fica compatível com a idealizada (problema rotineiro), gerando falsas expectativas no cliente;
- Não houve consolidação de fases, ou seja, iniciava-se uma fase antes mesmo de ter concluído a anterior. A pressa do cliente foi fator de peso para este problema ocorrer;
- Não houve troca entre programador e designer durante fases de desenvolvimento, após o trabalho ser repassado para fase da programação, não houve retorno ou consultas acerca de mudanças feitas de última hora pelo programador, ou ao menos para avaliação do designer.

Deste modo, a função do designer acabou sendo desvalorizada. O trabalho de layout acaba sendo uma ferramenta meramente estética e a opinião do profissional de design fica em segundo plano, mesmo que o layout tenha sido aplicado de forma mais fiel possível, a exclusão do mesmo nestas fases gera desmotivação pelo projeto como um todo.

Por fim, o software foi entregue ao cliente e uma série de testes foram realizados. Estes testes visam à validação do software sobre duas perspectivas: (1) a busca por erros de programação e bugs e (2) a verificação pelo cliente para validar se o que foi desenvolvido condiz com o que foi solicitado.

Com este processo de desenvolvimento de software podemos concluir que a união entre os profissionais de TI e de Design é indispensável para um bom projeto de desenvolvimento de software. Além disso, o uso de padrões e técnicas estabelecidos para a junção das áreas citadas auxilia a compreensão mutua dos profissionais envolvidos no projeto. Por fim, a qualidade do projeto de interfaces é muito superior em comparação com o projeto de interfaces desenvolvido apenas por profissionais de TI.

4.2.2 Análise De Projeto: App Para Ensino De Inglês A Crianças

Este projeto foi realizado nos anos de 2016 e 2017 para uma escola de inglês, com o objetivo de produzir um jogo digital voltado ao ensino de inglês para crianças. Ao final, o aplicativo deveria dar suporte ao ensino da língua inglesa por meio de música, componentes visuais e elementos de gamificação. As etapas iniciais (1 a 5) ocorreram no final de 2016 (considerando-se também o período de férias), e as etapas de desenvolvimento e entrega (6 a 14) ocorreram até abril de 2017. A equipe era composta por 4 integrantes: (1) gerente de projeto, (2) gerente de TI, (3) programador, e (4) designer de interfaces. As etapas do projeto são descritas a seguir:

- *Briefing:* Reunião com o cliente para o levantamento de requisitos;
- *Definição de Fases:* Elaboração do cronograma de trabalho, incluindo design conceitual, *sprints* de desenvolvimento e análise;
- *Reunião Kick-off:* Reunião da equipe para discussão dos requisitos e detalhamento do projeto. Definição da plataforma a ser utilizada (Android e IOS) e apresentação do cronograma;
- *Pesquisa Desk:* Pesquisa de materiais relacionados aos requisitos, para definição da estética e funcionalidades iniciais;
- *Definição de Tarefas:* Estabelecimento do sistema de troca de arquivos e guia de produção entre designer e programador;
- *Design Conceitual (Protótipo):* Desenvolvimento do protótipo conceitual para análise da equipe e do cliente;
- *Análise inicial:* Aprovação dos materiais gráficos e feedback por parte do cliente;
- *Design da User Interface (UI)*: Definição da estética final e detalhes técnicos de produção (exportação de gráficos, códigos de cores, grid da tela, linguagem correta para transferência de dados);

- *Desenvolvimento:* Desenvolvimento de todas as fases do jogo, respeitando as *sprints* estabelecidas. Cada *sprint* incluía geração dos componentes gráficos, programação e validação;
- *Validação:* Apresentação do progresso ao cliente, seguido de feedback do mesmo;
- *Correções:* Correção dos pontos levantados na validação e ajustes técnicos;
- *Testes:* Realização de testes alfa, seguidos de correções e ajustes;
- *Entrega:* Preparação dos arquivos finais (gráficos e código) para entrega ao cliente;
- *Governança:* Atividades relacionadas à gestão de projeto, incluindo: Reunião, Estabelecer Escopo, Estimativas, Preparar Ambiente, Analisar Viabilidade, Desenvolver Cronograma de Trabalho, Reunião Kick-off, Validar Requisitos, Definir Entregas, Definir Prazos e Custos, Gerenciar Qualidade, Gerenciar Comunicação.

Como se pode observar na Figura 13, o projeto inicia-se em novembro de 2016, pausa em dezembro para as férias, e retorna em janeiro de 2017, seguindo até a entrega no mês de abril. O gráfico demonstra as funções atribuídas a cada membro da equipe, incluindo o g*erente de projeto*, o *gerente de TI*, o *programador*, e o *designer de interfaces*. Como o projeto adotou uma metodologia de desenvolvimento ágil, muitas vezes as etapas se entrelaçaram, contrariando a previsão do cronograma.

Além disso, as iterações de desenvolvimento (incluindo desenvolvimento gráfico, programação e validação) repetiram-se variadas vezes, até que todas as fases do jogo estivessem finalizadas. O cliente esteve presente durante todo o processo de desenvolvimento para validar o progresso e fornecer feedback. Embora, como se pode notar pelo cronograma e pelas etapas descritas, o projeto tenha sido finalizado dentro do prazo e com resultado altamente satisfatório (de acordo com o cliente), é possível notar pontos baixos ao longo do processo.

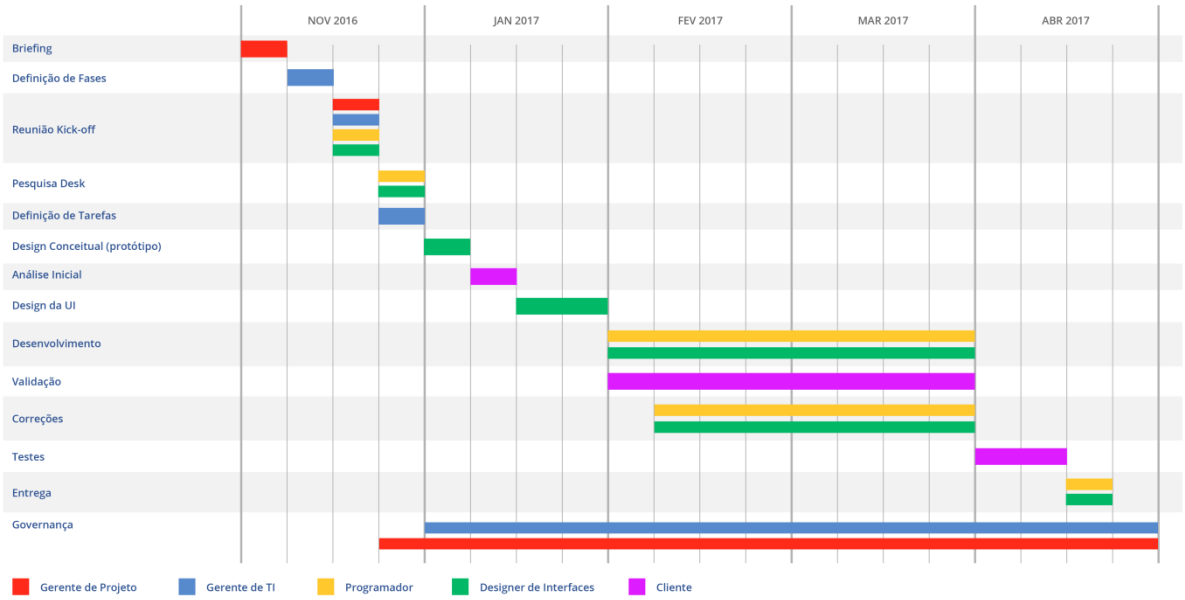

Figura 13: Cronograma app ensino de inglês

Dos problemas citados pelos profissionais envolvidos, tem-se:

- O levantamento de requisitos foi realizado pelo gerente de projeto, em conversa direta com o cliente, sendo que o restante da equipe não estava presente neste momento. No entanto, a presença de toda a equipe poderia extinguir a necessidade de apresentar o projeto aos membros posteriormente, evitando ruídos na comunicação (expectativas do cliente *versus* compreensão do gerente de projeto). Ademais, seria um momento interessante para que o designer extraísse informações sobre o direcionamento gráfico do projeto, permitindo a ele vislumbrar possibilidades e favorecendo a elaboração de *insights;*
- Estabeleceu-se que as trocas de arquivo entre design e programação se dariam por meio da alimentação de um repositório digital, e que se faria uso de um gerenciador on-line de tarefas para controlar o andamento das *sprints*. Reuniões adicionais foram realizadas por videoconferência em ocasiões esporádicas. Embora o aproveitamento destas ferramentas tenha sido, de forma geral, satisfatório, a comunicação remota nem sempre se mostrou efetiva, resultando em certos ruídos de comunicação que poderiam ser evitados;
- Em nenhum momento durante o projeto houve um canal de comunicação direto entre o cliente e o designer/programador. Toda a comunicação era centralizada pelo gerente de TI, que recebia informações de uma das partes e repassava a outra. Isso dificultou desnecessariamente a comunicação em momentos como o desenvolvimento gráfico e a validação, que poderiam ocorrer com maior rapidez;

- Outro problema identificado refere-se às validações internas após a liberação da programação em cada *sprint*. Nesse momento, os membros da equipe analisavam o progresso do projeto e faziam considerações para revisão. Entretanto, talvez devido aos ruídos ocasionados pela comunicação remota, tais considerações nem sempre eram devidamente revisadas. Algumas foram mesmo completamente negligenciadas, colocando em risco a qualidade final do software;
- Embora a equipe tenha realizado testes *alpha* internos, os testes *betas* ficaram sob a responsabilidade exclusiva do cliente. Desta forma, é plausível assumir que um valioso feedback do usuário final se tenha perdido, e consequentemente não foi incluído nas correções finais;
- A única documentação do projeto é referente às fases iniciais de preparação e levantamento de requisitos. Nenhum tipo de relatório foi emitido durante todo o processo de desenvolvimento nem após a conclusão do trabalho. Assim, fica praticamente impossível analisar com precisão os indicadores do projeto, bem como apontar necessidades de readequações e/ou adaptações de ordem gerencial e também operacional.

Apesar dos problemas apresentados, a avaliação geral do projeto é positiva. A equipe respeitou prazos e metas acordados com o cliente, que, por sua, vez, demonstrou estar satisfeito com o produto de software entregue. Para a equipe, todavia, não restam dúvidas de que solucionar tais problemas teria impactado positivamente no processo de desenvolvimento como um todo. Notam-se, sobretudo, falhas de comunicação que, por vezes, prolongaram as etapas ou geraram trabalho desnecessário, menosprezando, por conseguinte, as contribuições do design.

4.3 Questionários

O questionário foi desenvolvido baseado na seção 5.3 em pontos críticos apontados em pesquisas a respeito de problemas em desenvolvimento de software. O questionário completo consta no Apêndice A. Este questionário faz a conexão com a área de Design para identificar os possíveis problemas em um projeto que abrange as duas áreas. No total da pesquisa tem-se 75% das respostas de profissionais da área da TI e 25% da área de Design.

Quando questionado em relação a qual das fases de desenvolvimento (figura 14) de projeto acontecem mais falhas "fatais" ao projeto 75% dos participantes informaram que a fase inicial de Levantamento de Requisitos é mais crítica dentre as outras do

projeto, seguido por 16,7% que apontaram Análise e Design e 8,3% a fase de testes.

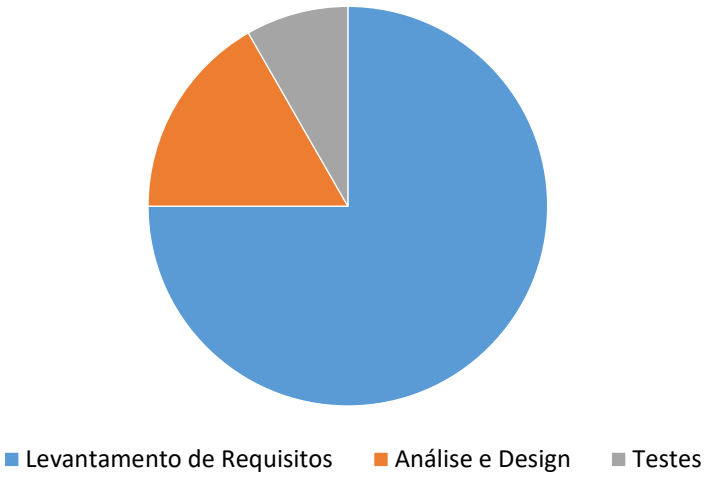

Figura 14: Fases de maior impacto

As opiniões predominantes foram as seguintes: o levantamento de requisitos normalmente é feito de modo incompleto e as pressas, pois muitas vezes uma única pessoa canaliza e define os requisitos, quando na verdade seria muito mais proveitoso se essa etapa fosse feita por toda a equipe integrando comercial, TI e Design. Esse levantamento geralmente equivocado e restrito feito a partir da visão de apenas um profissional prejudica todo o andamento do projeto. Além disso, as expectativas do cliente/usuário normalmente não são bem levantada ou registrada para o acesso de toda a equipe.

De forma geral apontaram para o fato de que um levantamento mal efetuado de dados, seja qual for o motivo, acarreta em decisões erradas e definição fraca de requisitos, criando efeito bola de neve de problemas que levam ao fracasso ou comprometimento do projeto. O termo requisitos é especialmente citado pela maioria como fator chave.

Quando questionados em relação aos principais motivos que levam um projeto fracasso 75% citaram falta de compreensão dos requisitos de projeto e Falta de planejamento; 66,7% apontaram a falta de comunicação do progresso individual e coletivo da dos membros da equipe; 58,3% citaram gerenciamento fraco ou inexistente e 41,7 % mencionam a inexistência de uma metodologia bem estruturada e apropriada ao projeto.

Na penúltima parte do questionário os participantes apontaram o nível de impacto de determinadas ações e questões dentro do projeto. De forma geral as de maior impacto são as questões relacionadas à comunicação, seja entre gerência e equipe,

ou entre os membros da mesma, na forma de feedback, contextualização projetual ou troca de ideias que acarretam na definição fraca de requisitos, inabilidade de detectar possíveis erros e riscos, e por fim falta de comprometimento da equipe em um projeto com gerenciamento e processos fracos.

Dos pontos específicos considerados de alto impacto tem-se:

- Falta de interação e comunicação na equipe;
- Gerência fraca;
- Falta de expertise técnica;
- Falta de conhecimento de gerenciamento de projeto;
- Inabilidade de Detectar problemas antecipadamente no produto e no processo;
- Falta de conhecimento da área do colega;
- Falta de domínio de linguagem descritiva por parte do designer para construção de interface em código.

Quando questionados se há necessidade de participar de discussões do projeto que vão além do seu domínio e se isso ocorre com frequência, foi unânime que é indispensável participar de todas as discussões pertinentes e críticas ao processo, porém muito afirmam que não é necessário aprofundar em questões que estão muito além da área, focando em pontos essenciais que requer a atenção de todos os membros, aqui acentuam a tarefa de um gerência forte que saiba criar pauta adequada para as reuniões da equipe. Agora em relação à realidade, muitos afirmaram que é comum não haver esforço da equipe em transmitir ideias fora do domínio de determinados integrantes e que isso afeta o desenvolvimento.

Por fim foi perguntado, quais os desafios que cada profissional acredita existir em projetos multidisciplinares. Como era de se esperar os profissionais da TI apontam para a falta de conhecimento técnico básico de programação do designer, que acarreta em propostas desalinhadas com os requisitos do projeto. Em contrapartida os designers apontam para a falta de encorajamento e interesse criativo da parte dos programadores em detrimento a solução prática e analítica de um problema.

Para encerrar abaixo fica uma observação bastante pertinente ao assunto feita por um dos participantes: “A principal dificuldade que eu percebo entre os trabalhos do designer e do analista/programador é a busca do *ponto de tolerância*. Este ponto é tênue e difícil de perceber. Até que ponto um designer pode (ou deve) alterar um logotipo para ser mais fácil de ser mapeado e programado pelo programador? ”.

5 RESULTADOS

5.1 GAIA Agile Framework

O primeiro passo para reenquadrar o framework atual ao modelo ágil foi definir as 5 grandes áreas de acordo com as atividades e função exercidas de modo a facilitar a distribuição das mesmas de forma iterativa. Na tabela 9 foram organizados os requisitos levantados até o momento (fundamentação teórica, análise do framework, relatório de projetos reais e questionários) de todas as questões que devem ser abordadas no novo framework.

Tabela 9: Parâmetros definidos

Requisito	
Engenharia Software	
Modelo de Framework	Agile
Base	Scrum
Ciclo de Vida	Iterativo Incremental
Foco	Comunicação
Gestão de Design	
Abordagem	Design Thinking
Base	Ferramentas de DT
Foco	Inovação, Comunicação e Criatividade
Gameficação	
Abordagem	Análise de Motivação e Objetivos
Foco	Feedback de Desempenho
Análise GAIA	
Pontos Críticos	Necessidade de Testes, Inserção do Design, Delegação de Decisão, Gerar Documentação.
Questionários	
Pontos Críticos	Maior foco no levantamento de Requisitos e Comunicação.

Cada área ou etapa definida apresenta atividades que podem ser distribuídas conforme as necessidades ao decorrer de um projeto. Conforme a figura 15 tem-se:

- Análise Inicial e Governança: Abrange as atividades iniciais de projeto assim como delimita o trabalho da governança que permeia e abrange todo o processo. As atividades incluem: Reunião, Estabelecer Escopo, Estimativas, Preparar Ambiente, Analisar Viabilidade, Criar Project Charter, Reunião Kick-off,

Expandir WBS, Validar Requisitos, Definir Entregas, Rastreabilidade, Governança (que engloba Planejar Indicadores, Recursos, Análise de Riscos, Prazos e Custos, Revisar Planos, Análise Viabilidade), Gerenciar Qualidade, Gerenciar Comunicação;

- Análise e Design: Abrange as atividades de levantamento de requisitos, revisão, elaboração de cronograma, Definição de Tarefas, Planejamento das Fases seguintes e Testes;
- Execução e Implementação: Abrange as atividades da parte operacional do projeto como Execução de Tarefas, Realização de Correções, Prototipação e Implantação de Resultados;
- Teste: Área dos testes unitários, Análise de resultados e Testes de Integração;
- Entrega: Finalização de Projeto.

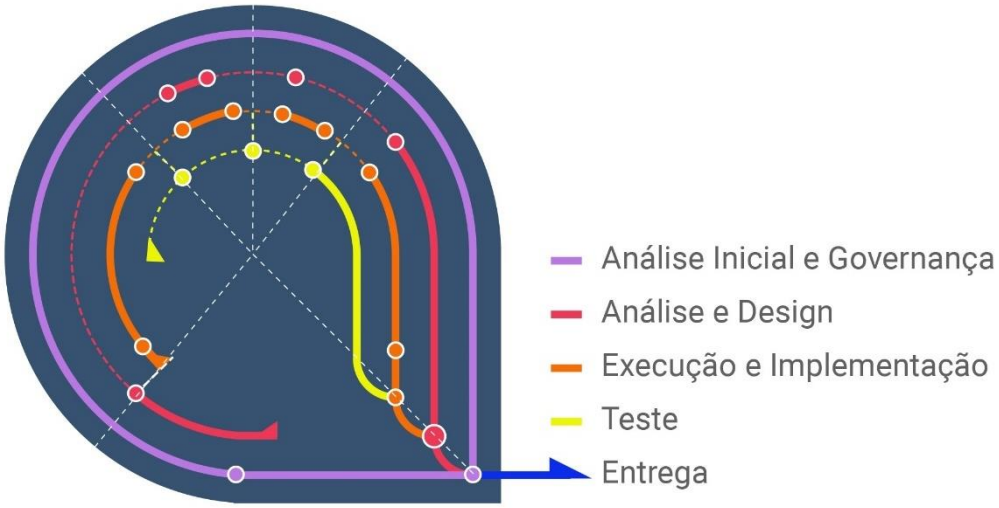

Figura 15: GAIA agile

O processo possui natureza cíclica e iterativa. Na figura 16 foram demarcadas as diferentes zonas de iteração, ciclos fechados de desenvolvimento, para cada etapa do projeto definida na Análise Inicial e Governança.

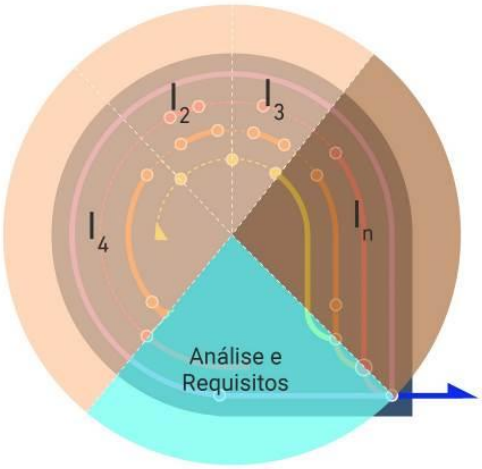

Figura 16: GAIA iterações

Na tabela 10 pode-se visualizar a distribuição das fases dentro de cada área de desenvolvimento e profissionais envolvidos em cada uma. Foi dado ênfase no engajamento maior de todos os integrantes ao decorrer das etapas para corrigir problemas mencionados anteriormente.

Tabela 10: Fases GAIA agile

Fase	Etapa	Integrantes	Identificação
Análise Inicial e Governança	Reunião	Gerente e Cliente	A1
	- Estabelecer Escopo		
	- Estimativas		
	Preparar Ambiente		A2
	Analisar Viabilidade		A3
	Criar Project Charter	Equipe e Cliente	A4
	Reunião Kick-off	Equipe	A5
	Expandir WBS	Gerente	A6
	Validar Requisitos	Equipe	A7
	Definir Entregas	Gerente	A8
	Rastreabilidade		A9
	Governança	Gerente	A10
	- Planejar Indicadores		
	- Recursos		
	- Análise de Riscos		
	- Prazos e Custos		
	- Revisar Planos		
	- Análise Viabilidade		
	Gerenciar Qualidade	Equipe	A11
	Gerenciar Comunicação	Equipe	A12

Tabela 10: Continuação da página anterior

Análise e Design	Levantar Requisitos	Equipe	D1
	Revisar Requisitos		D2
			D3
Análise e Design	Elaborar Cronograma		D4
	Elaborar Tarefas	Gerente	D5
	Planejar Fases e Testes		D5
Execução e Implementação	Executar Tarefas	Equipe	E1
	Protótipo		E2
	Realizar Correção		E3
	Implantar Resultados		E4
Teste	Executar Teste Unitário	Equipe, Usuário	T1
	Análise de Resultados	e Cliente	T2
	Executar Teste de Integração		T3
Entrega	Entrega Final	Equipe e Cliente	E1
	Reunião de Feedback		E2
	Confraternização		E3

Utilizando os números de identificação foi construído um modelo visual de operação do framework (figura 17). Neste modelo é possível perceber o ciclo de vida de cada fase e como elas se intercalam e repetem ao longo de todo o processo. Nota-se uma ênfase maior no estágio de análise e levantamento de requisitos, que como foi apontado anteriormente é extremamente essencial para o bom funcionamento de um projeto.

As atividades de governança como gerenciamento de comunicação e qualidade são extraprocessuais, pois ocorrem initerruptamente ao longo de qualquer projeto e é executado por toda a equipe sob o comando do gerente.

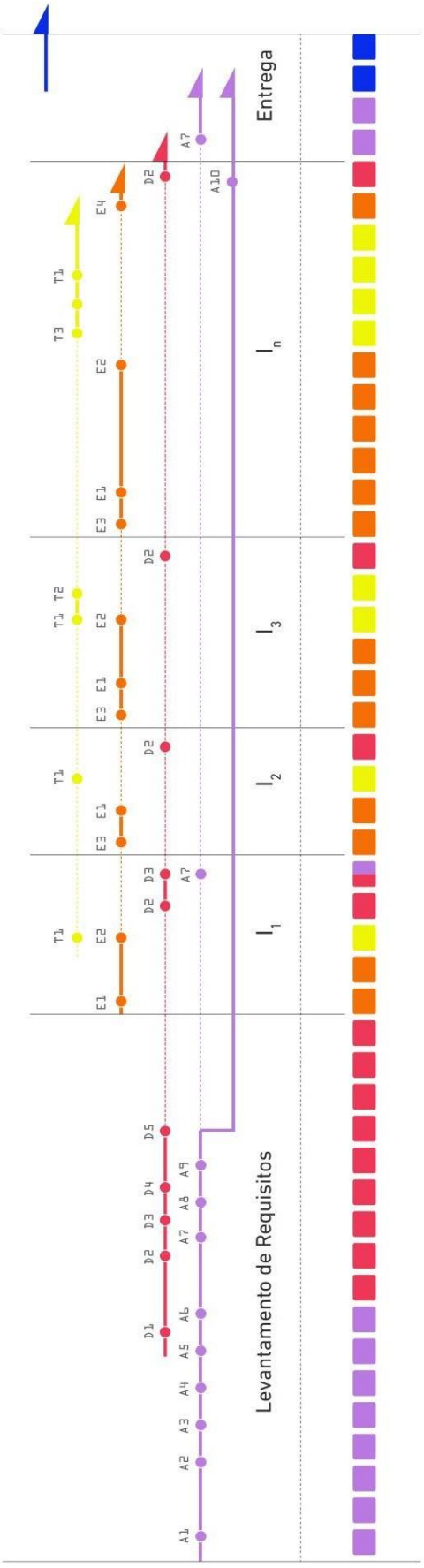

Figura 17: GAIA agile framework

O número de iterações não é fixo, sendo definido pela equipe na fase inicial e podendo ser estendida ou reduzida conforme as necessidades do projeto. Atividades de governança permeiam todo o projeto, ela é o olhar macro que cuida de internos e externos do projeto. Quais atividades de governança serão desempenhadas dependerá do tipo de trabalho, e como o gerente irá desempenhá-las depende totalmente das decisões tomadas inicialmente. Ao final de cada iteração torna-se obrigatório a realização de reuniões de equipe para troca de ideias, atualização do desempenho e produção individual e definição das etapas seguintes.

As fases de análise e design, execução e testes trabalham de forma cíclica e intercalada, desenvolvendo-se pequenas porções definidas inicialmente, analisando o trabalho feito e validando por meio de testes que guiarão a fase seguinte que se repete até todos os requisitos definidos sejam atendidos. Por fim a governança encerra o trabalho com coleta da documentação produzida pela equipe ao longo do trabalho e feedback de desempenho e a entrega é efetuada.

5.2 Convergindo Design Thinking e Gaia Agile Framework

Definido o framework GAIA dentro dos padrões Agile resta agora delimitar as ferramentas de design apropriadas a cada fase e definir a jornada do design e designer dentro do fluxograma da metodologia, assim como as atividades desempenhadas por cada profissional e a zona de convergência (figura 18) e compartilhamento de insights entre as duas áreas.

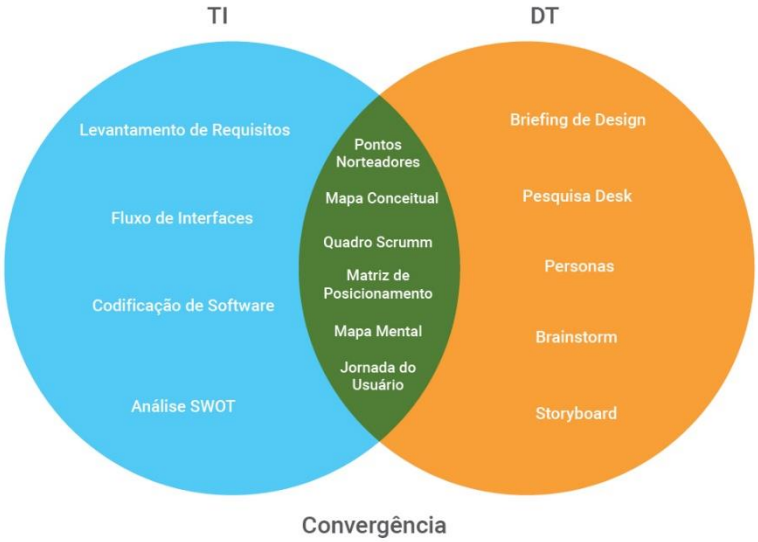

Figura 18: Convergência TI e DT

Abaixo na tabela 11 foram reunidas de forma resumida as ferramentas de design thinking e IHC que foram integradas ao framework, de acordo com a etapa de desenvolvimento em cada fase, para potencializar o projeto e reduzir obstáculos no decorrer do trabalho.

Tabela 11: Ferramentas de design thinking

Estágio de Desenvolvimento	**Ferramenta de Design**	**Fase GAIA Agile**
Imersão Preliminar	Briefing	A5 – Reunião Kick-off
	Pesquisa Desk	D1 – Levantamento de Requisitos
	Pesquisa Exploratória	D1 – Levantamento de Requisitos
	Reenquadramento	D1 – Levantamento de Requisitos
Imersão em Profundidade	Entrevistas	*D1 – Levantamento de Requisitos*
	Cadernos de Sensibilização	*D1 – Levantamento de Requisitos*
	Sessão Generativa	*D1 – Levantamento de Requisitos*
	Um dia na Vida	*D1 – Levantamento de Requisitos*
	Sombra	*D1 – Levantamento de Requisitos*
Análise e Síntese	Cartões de Insight	D2 – Revisão de Requisitos
	Diagrama de Afinidades	D2 – Revisão de Requisitos
	Mapa Conceitual	D2 – Revisão de Requisitos
	Critérios Norteadores	D3 – Validar Requistos
	Personas	D2 – Revisão de Requisitos
	Mapa de Empatia	D2 – Revisão de Requisitos
	Jornada do Usuário	D3 – Validar Requisitos
Ideação	Brainstorming	
	Workshop de Cocriação	D5 – Elaborar Fases
	Cardápio de Ideias	D5 – Elaborar Fases
	Matriz de Posicionamento	D4 – Matriz de Posicionamento
Prototipação	Protótipo em Papel	E1 – Protótipo
	Modelo de Volume	
	Encenação	
	Storyboard	E1 – Protótipo
Teste	Análise Heurís't'ica	T1 – Teste Unitário
	“Um dia na vida”	T3 – Teste de Integração
	Teste Usuário	
	Sombra	
Documentação e Feedback	Caderno de Projeto	Governança

As atividades foram distribuidas de acordo com as fases definidas na seção anterior e têm o objetivos de potencializar cada fase, fomentando *criatividade*, *comunicação*, *insights* e *organização* das informações produzidas para melhor visualização da equipe. Cada ponto em verde na figura 19 representa a ação do design e/ou uso da ferramenta de design thinking.

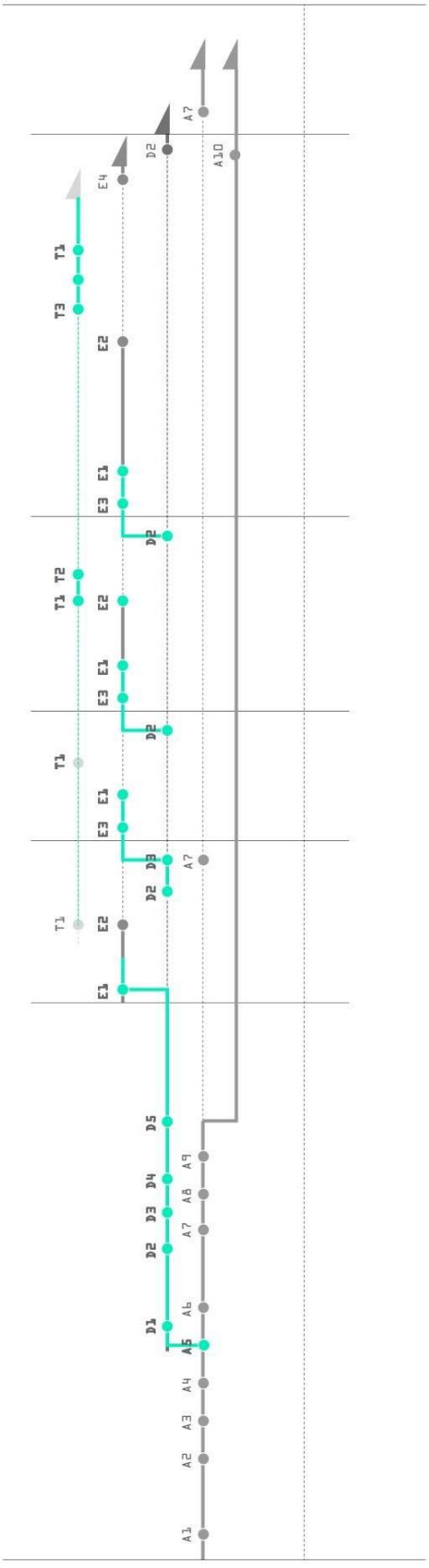

Figura 19: Função do design

Na figura 20 tem-se detalhado as etapas do processo com as ferramentas de Design empregadas em cada uma.

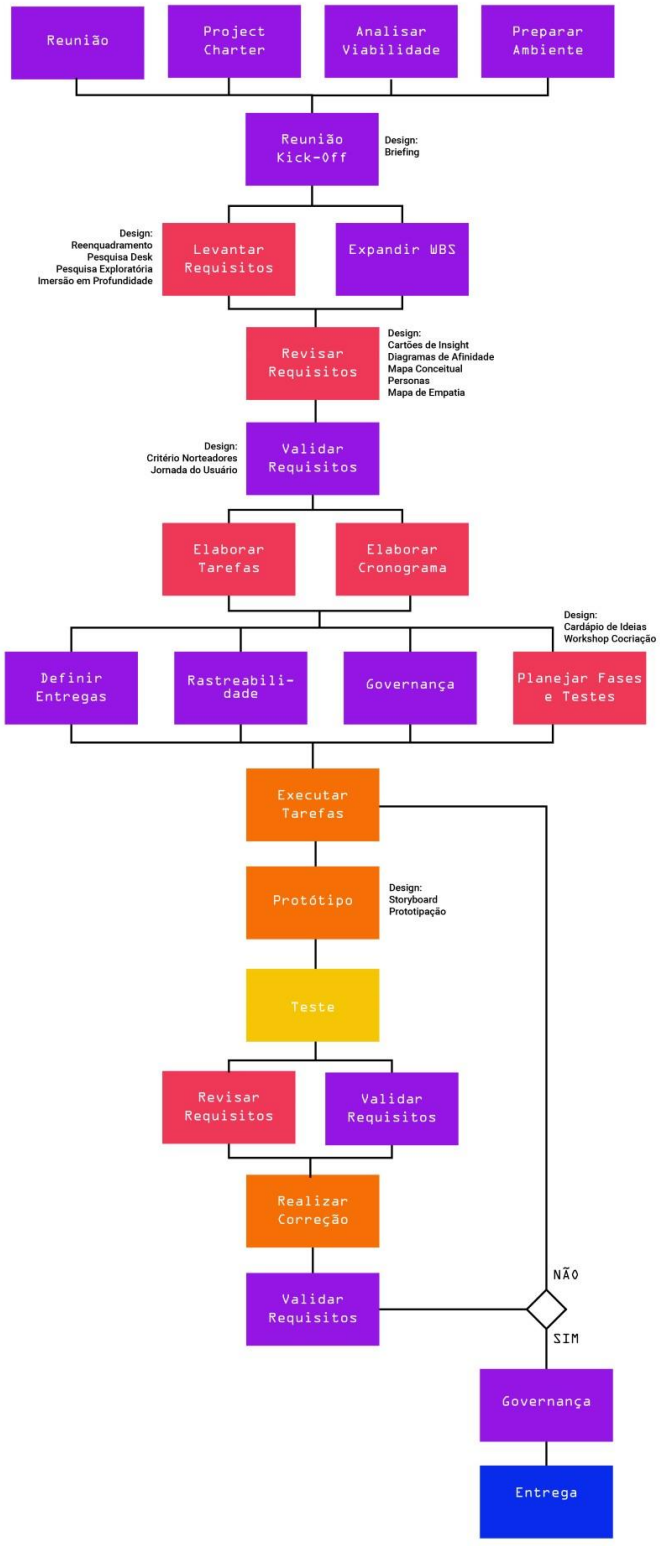

Figura 20: GAIA framework detalhado

5.2.1 Detalhamentos das Ferramentas

Nesta etapa será realizado o detalhamento das ferramentas e proposta de alguns modelos alinhados a realidade da produção de software, mais especificamente as necessidades de projeto do laboratório GAIA.

Briefing: Cada projeto possui suas peculiaridades, porém há questões essenciais inerentes a área de cada atividade. Para um briefing mais completo em projetos desta natureza que aborde todos os problemas mencionados é importante levantar os seguintes dados do cliente e também, uma vez que a equipe é nova e temporária, um briefing interno minimizará problemas ao decorrer do trabalho (tabela 12).

Tabela 12: Briefing

Para o Cliente	**Entre Programador e Designer**
Que tipo de software você tem em mente?	Quais tipos de projeto você já trabalhou?
Que público pretende atender?	**Exemplos de trabalhos desenvolvidos.**
Qual a necessidade deste público?	Qual linguagem de software que tem conhecimento?
Qual a sua necessidade em relação ao software?	**Qual conhecimento que um tem da área do outro?**
Possui uma empresa? Está empresa possui identidade visual?	Que tipo de linguagem descritiva é apropriada para transmissão de dados entre ambas as partes?
Quais suas expectativas e referências quanto ao resultado do software?	
Escopo e Orçamento esperados?	
Já existe alguma versão disponível e operando no mercado?	**Quais especificações técnicas devem ser evidenciadas (programador tem conhecimento de softwares de edição de imagem, designer tem conhecimento básico de código)?**
Se sim, quais os problemas existentes que devem ser corrigidos?	
O que deve ser evitado?	Qual seu método de trabalho?
	Quais canais de comunicação são mais confortáveis para cada um?

Em fases iniciais um Briefing bem realizado com a participação de toda a equipe além de eliminar dúvidas em relação ao projeto, serve também como uma ferramenta para informar a todos do contexto do cliente, dos requisitos do software e principalmente a formação da dinâmica de equipe que guiará o processo de produção.

Pesquisa Desk e Exploratória: Como uma pesquisa desk deve ser desenvolvida

é muito subjetivo e diz respeito ao método que cada indivíduo tem de se preparar para um novo desafio que deve ser resolvido. Portanto delatar um trajeto de pesquisa se torna fútil, porém pode-se atentar a quem pesquisa de dados importantes que devem ser atentados em um projeto de desenvolvimento de software como:

- Pesquisa sobre o histórico do cliente;
- Pesquisa sobre usuário para construção de personas, mapas de empatia e jornada do usuário;
- Pesquisa de mercado e similares;
- Pesquisa sobre ferramentas que serão utilizadas para produção do software;
- Pesquisa sobre estética que será abordada para fundamentar a criação.

Reenquadramento | Debriefing: Após o levantamento de dados e informações projetuais o reenquadramento oferece a chance de discussão aprofundada com o cliente para reavaliação de questões chaves do projeto, uma vez que as ideais e requisitos serão analisados com base em dados mais consolidados e novas questões ou pontos de vistas podem ser explorados com o uso desta ferramenta. A máxima desta ferramenta é revisitar um lugar familiar com olhar amadurecido.

Imersão em Profundidade: As ferramentas de imersão em profundidade podem ser utilizadas sempre que for necessário aprofundar mais no universo do usuário/cliente, o que irá depender dos requisitos específicos de cada projeto. Por meio de entrevistas, atividades planejadas e guiadas, construção de mapas e cadernos de motivação é possível delimitar um perfil ou perfis de usuário e suas necessidades/expectativas que guiarão o processo em um caminho mais definido sendo possível prever erros e contorna-los antecipadamente.

Cartões de Insights: A partir das informações coletadas até o momento, a equipe deve organizar estas informações em cartões separados por tema de insights e conceitos chaves que são pertinentes ao projeto. Estes cartões podem ser organizados em grupos de ideias *(Diagrama de Afinidades)* que se relacionam de modo a prover uma visão global dos requisitos e ideias até o momento propiciando assim insights para solução e definição de parâmetros de produção.

Mapa Conceitual: No mapa conceitual (figura 21) devem ser organizadas todas as informações relevantes para o projeto, o mapa ajuda a visualizar os conceitos chaves do projeto, nele pode ser incluso informações do usuário e os diagramas de personas assim como a jornada simplificada do usuário, a adição do quadro de atividades Scrum

é bastante recomendada por fornecer uma visão ampla das atividades definidas pelo programador.

Está é uma ferramenta chave do projeto, todas as informações levantadas e relevantes devem ser direcionadas a ela. A partir dela que se começa o processo de filtragem e hierarquização dos critérios norteadores do projeto e partir para a fase de ideação.

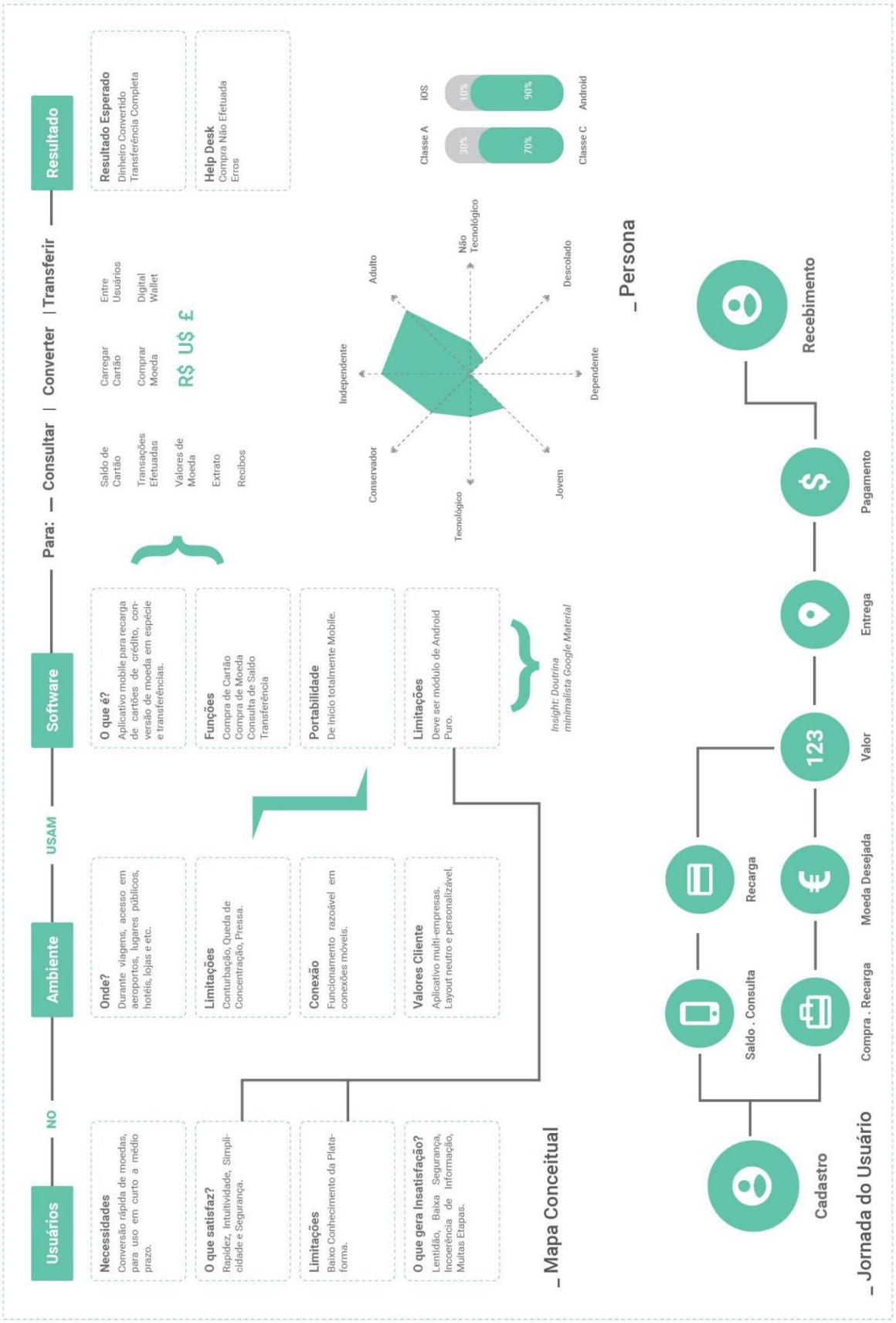

Figura 21: Mapa conceitual

Personas: Por meio do briefing, pesquisas desk, entrevistas ou grupos focais são possíveis determinar os tipos de usuário que serão alvo de um software. Alguns apresentam um tipo muito específico de usuário que requer nível de pesquisa mais aprofundada, como por exemplo, em trabalhos de acessibilidade que lida com públicos com limitações específicas. Outros possuem vários usuários alvos e é necessário um levantamento mais genérico e abrangente dos diferentes perfis e como estes afetam a jornada e experiência do usuário como software.

Os cartões de insights são de grande ajuda quando é preciso enquadrar os usuários em diferentes categorias, por propiciar uma visão fragmentada de atributos que podem ser facilmente alocadas e organizadas. O que se deve ter em mente é identificação de pontos críticos e características de alto impacto na produção, e a sumarização destes atributos em gráficos polarizados (figura 22), que ajudam a determinar quais os níveis de impacto de cada grupo.

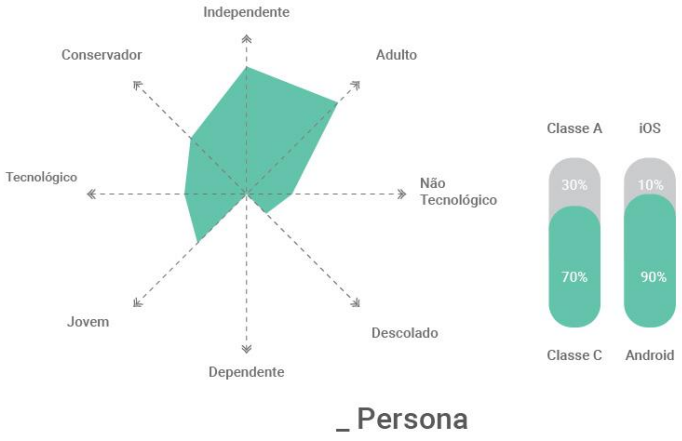

Figura 22: Personas

Critérios Norteadores: Os critérios norteadores fundidos com Scrumm Board (figura 23) é a hierarquização dos requisitos chave que devem ser solucionados. A ordem de importância deve sempre estar adequada as iterações do projeto, ou seja, não devem ser colocadas no topo os requisitos mais importantes uma vez que dentro de uma iteração deverá constar elementos complementares que são necessários para apresentação parcial de um software em execução.

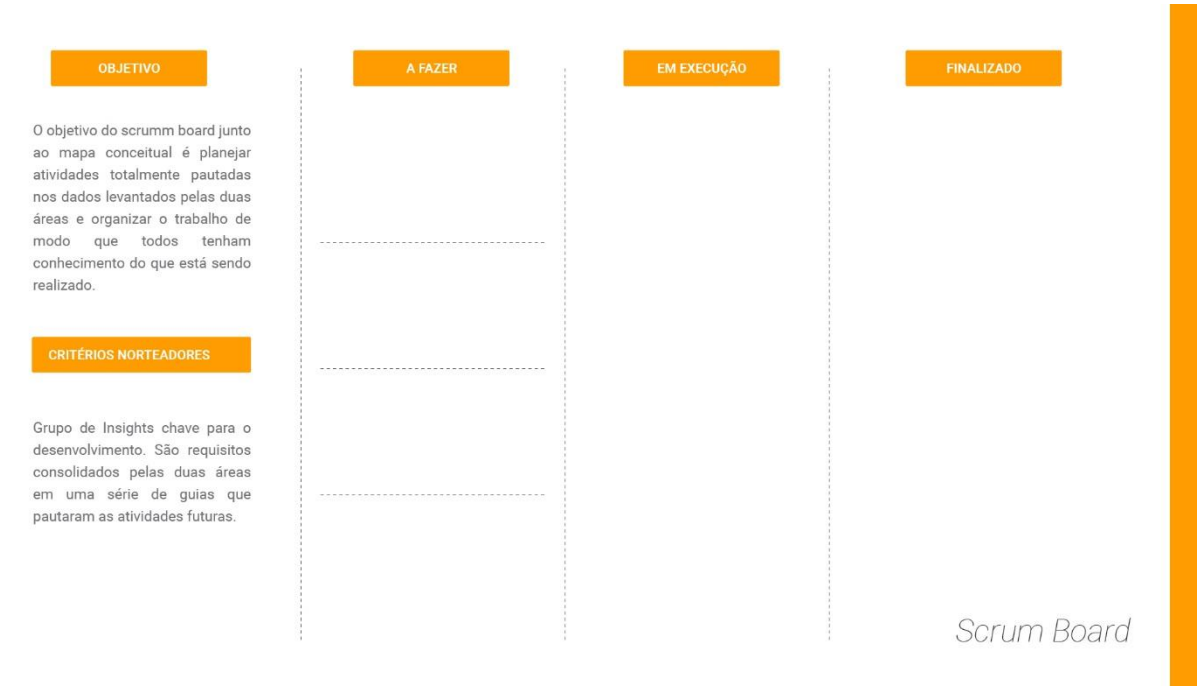

Figura 23: Scrum board

Jornada do Usuário: A jornada do usuário (figura 24) deve ser um gráfico visual de todas as formas e etapas de interação que uma pessoa terá com determinado software. Nele é importante evidenciar expectativas do usuário, possíveis erros e como contorna-los. Fatores internos quanto externos devem ser abordados e registrados aqui, desde o primeiro contato com o produto (que pode incluir até mesmo a parte de marketing e divulgação) até a interação final com o mesmo.

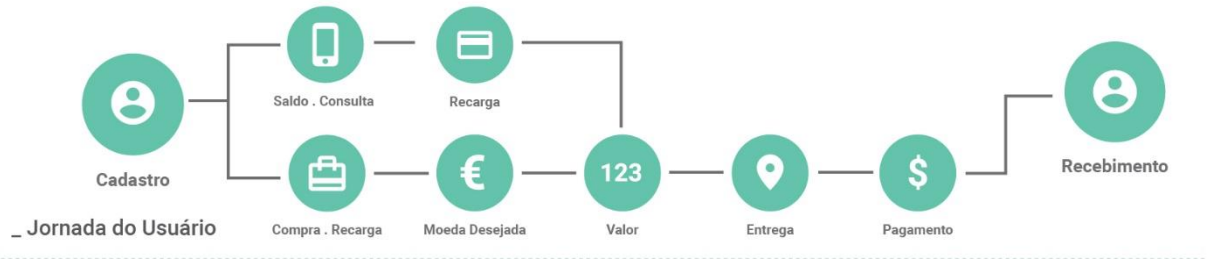

Figura 24: Jornada do usuário

Para projetos de software é extremamente interessante a junção da jornada do usuário com o fluxo de telas (figura 25) produzido pelo programador, pois permite construir um passo-a-passo completo de todas as tarefas que serão executadas.

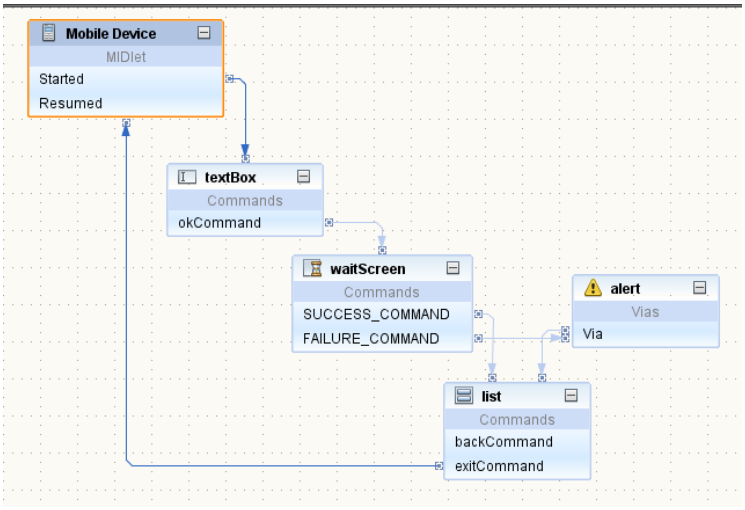

Figura 25: Fluxo de telas

Storyboards: O storyboard (figura 26) é uma ferramenta excelente para visualização inicial de atividades, por ser desenvolvido todo em papel é um momento onde designers e programadores podem trabalhar em mesmo nível técnico e expor suas ideias de forma clara e rápida, e como resultados têm um mapa de telas completo que servirá de guia para as fases de desenvolvimento.

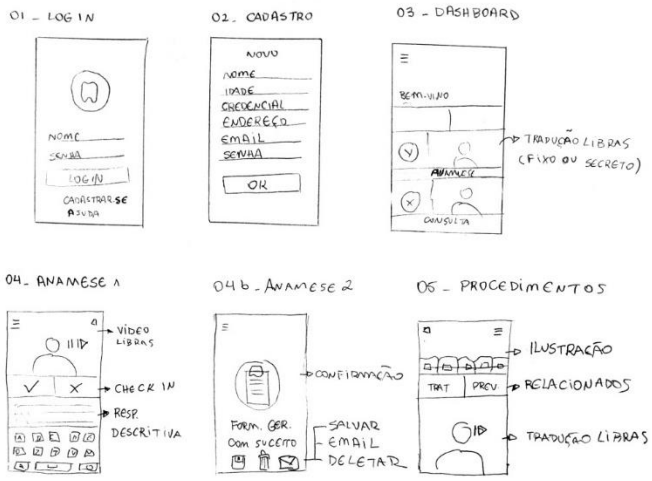

Figura 26: Storyboard

Análise Heurística Guiada: A análise heurística é uma série de testes internos que podem ser aplicadas pela própria equipe ou por um especialista em usabilidade. Na figura 27 foi proposto um modelo expandido da análise com questões guias que devem ser feitas ao longo de todo projeto, especialmente ao fim de cada iteração. O teste permite a finalização de fases mais consolidadas e alinhadas com as expectativas do cliente e usuário, além de ser uma ótima alternativa quando não se é possível testar com o usuário real em estágios iniciais.

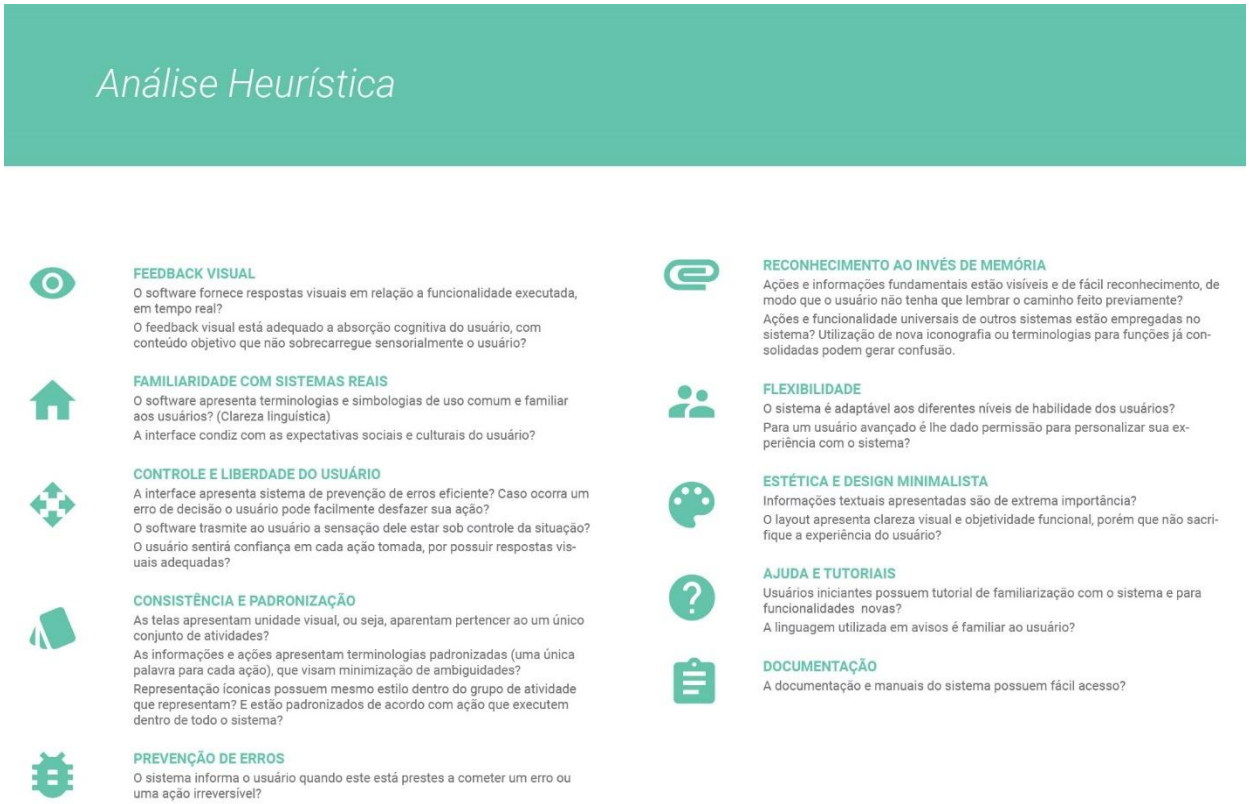

Figura 27: Análise heurística guiada

Caderno de Projeto: A documentação do processo e suas soluções é uma tarefa que deve ser desenvolvida ao longo de todo projeto pela equipe e organizada pelo gerente. Um projeto bem documentado torna-se fonte preciosa de informações de desempenho, assim como aponta fraquezas dentro do processo e na equipe que precisam ser melhoradas. Para os integrantes a documentação é uma ferramenta que poderá ser reutilizada para amadurecimento profissional assim como para consulta de dúvidas recorrentes e maneiras de trabalhar mais adequadas a realidade do laboratório. Novas equipes que se formam podem estudar estes relatórios como ponto de partida para nivelamento e briefing entre profissionais.

5.2.2 Roda de Ferramentas

Para organizar e ajudar a equipe a visualizar a ordem natural e lógica das ferramentas em relação ao projeto foi desenvolvida uma roda (figura 28) interativa que mostra qual ferramenta é mais apropriada para cada fase e sua ordem linear. Porém essa ordem não deve ser encarada como obrigatória uma vez que as ferramentas podem e devem ser revisitadas e atualizadas constantemente ao longo de qualquer projeto.

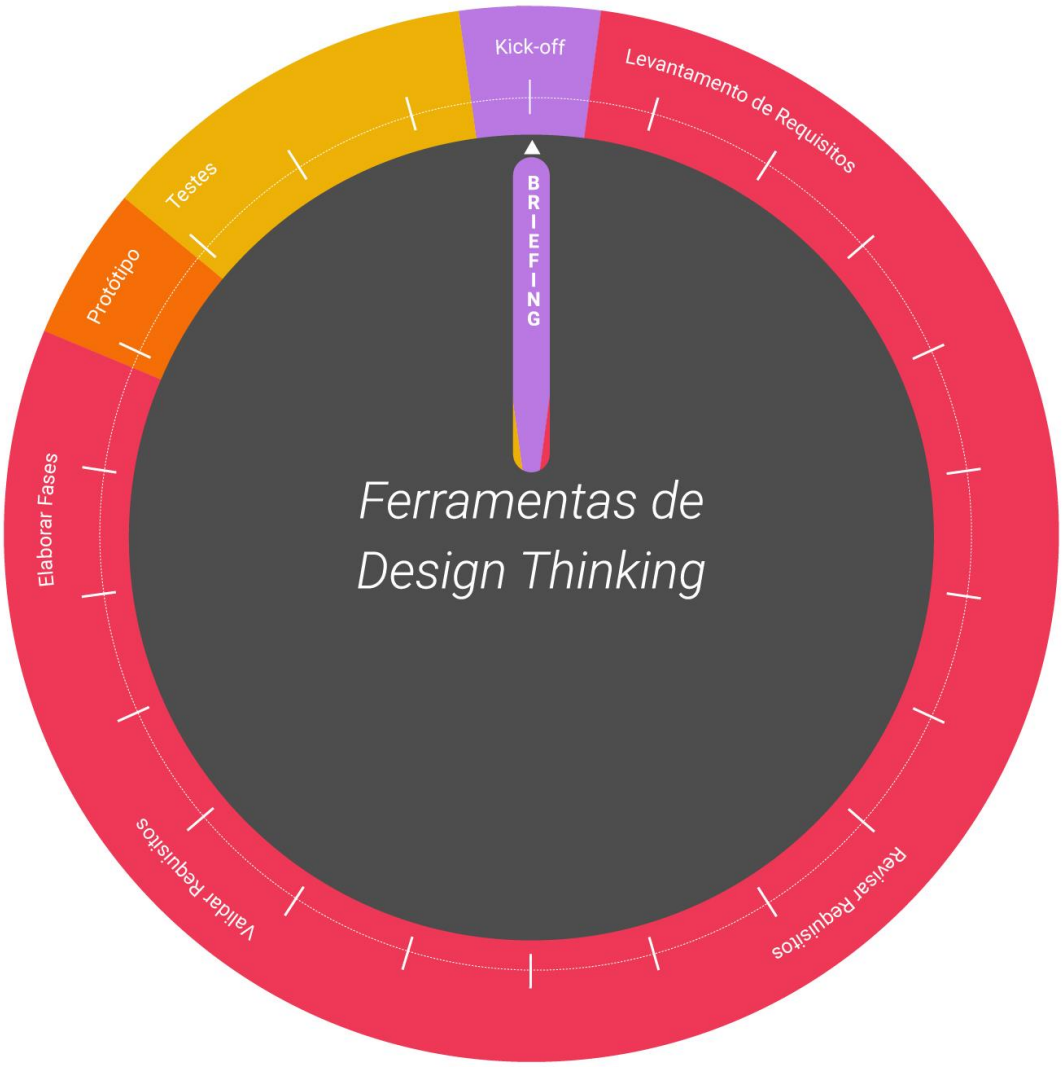

Figura 28: Roda de ferramentas

5.3 Questionário com Especialistas

Como forma de validação, no sentido de encontrar um indicativo de que a proposta adotada de framework possui potencial para minimizar os obstáculos no desenvolvimento de software, foi realizada uma análise com especialistas das duas áreas. Primeiramente foi apresentado o modelo desenvolvido de forma detalhada, assim como os dados levantados que serviram de base para a pesquisa, para os dois tipos de especialistas, da área de design e outros da área da computação. A avaliação busca aceitação de pelo menos 90% como indicativo de potencial da proposta.

No total foram consultados 10 especialistas, sendo 4 da área da computação e 6 da área de design, todos ex-alunos participantes de projetos de desenvolvimento de software e atuando na área atualmente. Para mensurar a avaliação dos especialistas,

foram apresentados questionários (Apêndice B) com 9 afirmações voltadas a aspectos críticos do framework e pedido que dessem uma nota. A escala de notas terá índices que variam de 1 a 5. Sendo que 1 representa a expressão "discordo plenamente" e 5 representa "concordo plenamente". Para a avaliação espera-se encontrar nota igual ou maior à 4.5 para ser considerado um bom índice de aceitação e potencial do framework proposto. A tabela 13 apresentam os dados obtidos com essa aplicação.

Tabela 13:Resultados da aplicação do questionário qualitativo

Participante	**Perguntas | Notas**									**Média**
x	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	x
1	5	5	4	4	5	4	4	4	5	4,4
2	5	4	5	5	4	5	5	4	5	4,6
3	5	5	5	4	4	5	5	5	5	4,7
4	5	5	4	5	5	4	4	5	5	4,6
5	5	5	4	5	5	5	5	5	5	4,8
6	5	5	5	5	5	5	5	5	5	5
7	5	5	4	5	5	4	5	4	5	4,6
8	5	4	5	5	5	4	5	4	5	4,6
9	5	4	5	5	5	5	5	5	5	4,8
10	5	5	5	5	5	5	5	5	5	5
									TOTAL	4,7

De acordo com a tabela 13, tem-se uma média de 4,7 de aceitação das afirmações proposta no questionário com base no trabalho proposto, sendo assim o índice de aprovação é maior que 94%, indicando o potencial do modelo para as duas categorias de participantes. Portanto o framework ágil de convergência GAIA pode passar à próxima fase e ser aplicado no ambiente de trabalho para obter melhores resultados uma vez que conta com além de um bom embasamento teórico, a aprovação dos especialistas envolvidos.

6 CONCLUSÃO

Uma vez que um projeto de desenvolvimento de software envolve design e suas diversas faces (design de UI, UX, usabilidade, ergonomia, etc), engenharia de software e IHC, não há motivos para dar preferência a uma disciplina em detrimento das outras, colocando-as como secundárias no impacto que possuem em um projeto.

Além de ser um guia para formatação de novos frameworks para pequenas empresas ou equipes de desenvolvimento, o trabalho apresenta de forma consistente a compilação de dados essenciais como: principais obstáculos em projetos desta natureza, controle de qualidade, convergência de disciplinas e equipes multidisciplinares, fomento da criatividade, entre outros.

Com o levantamento bibliográfico foi possível levantar os fatores cruciais para o estabelecimento de um framework, como ciclos de vida e suas vantagens, metodologias ágeis e como operam dentro de um processo de desenvolvimento de software e ferramentas de organização da TI. Da parte do Design foram levantados os tipos de impacto de gestão (operacional e estratégico), a abordagem do design thinking para o processo de criação e suas ferramentas.

Por meio da análise de similares foi estudado o perfil de criação da empresa Google e retirado parâmetros que guiaram a definição do novo modelo para o laboratório GAIA, assim como a proposta de um repositório/documentação tendo como referência o Google Material, que como visto é uma tentativa extremamente bem-sucedida de fundir design e programação em uma só linguagem de criação.

A análise do problema levantado no presente trabalho estabeleceu o modelo de framework convergente discutido no capítulo 3.4, por meio deste determinou-se um processo capaz de englobar as atividades de TI junto as de Design de Interface dentro de um contexto dinâmico, com foco na distribuição do poder de tomada de decisão entre os membros da equipe. Todas ferramentas levantadas no capítulo 3.2 foram integradas ao modelo GAIA de produção para organizar e padronizar o processo de criação e fomentar a geração de insights.

Todos os resultados foram pautados em uma extensa pesquisa teórica e estudos de casos reais, questionários e análises internas do laboratório, que apontassem na direção mais correta para formatação de uma metodologia de trabalho mais compatível com a realidade. Partindo da questão deflagrada no início do projeto de como criar um

framework de convergência de disciplinas, não só se chegou ao resultado com um framework estruturado, como também um guia que pode ser utilizado para analisar outros processos de criação de software.

A pesquisa passou por uma avaliação com especialistas de modo a validar o estudo feito para a criação do modelo proposto e o seu potencial otimizador. Ao que indica os resultados, o framework teve alta índice de aprovação e pode ser implementado no laboratório a fins de obter validação final de sua eficácia e eficiência, assim como dados para a próximas fases evolutivas do modelo, que nunca deve parar de se adaptar.

Por fim, o trabalho levanta questões relevantes para fomentar pesquisas futuras, como por exemplo:

- Análise do custo/benefício da aplicação de um framework abrangente dentro do ambiente corporativo;
- Aplicação do framework em diversos ambientes para levantamento de dados concretos para validação do mesmo e sua possível evolução;
- Estudo das dinâmicas e técnicas das duas áreas na fusão destas em ferramentas mais abrangentes;
- Proposta de repositório digital para laboratórios de TI para levantamento e armazenagem de documentação gerada em projetos;
- Análise da motivação interna/externa dos profissionais, sob diversas óticas, podendo contar com casos específicos, como o laboratório GAIA.

REFERÊNCIAS

[1] ACUÑA, S. T.; GÓMEZ, M. N.; HANNAY, J. E.; JURISTO, N.; PFAHL, D. Are team personality and climate related to satisfaction and software quality? Aggregating results from a twice replicated experiment. *Information and Software Technology*, v. 57, p. 141–156, 2014.

[2] BARROS, V. T. O. *Avaliação da interface de um aplicativo computacional através de teste de usabilidade, questionário ergonômico e análise gráfica do design.* Dissertação (Mestrado) - UFSC, Florianópolis, 2005.

[3] BARTOLOMEU, B. M. P. *Design thinking indústria de IT: implementação e adoção: o caso da Novabase,* 2014. Tese de Doutorado, Instituto Superior de Economia e Gestão.

[4] BEZERRA, E. *Sistemas UML.* 3ª ed. Rio de Janeiro: Elsevier, 2014.

[5] BROWN, T. *Design Thinking*: uma metodologia poderosa para decretar o fim das velhas ideias. Rio de Janeiro: Elsevier, 2010.

[6] CARVALHO, J. O. F. *O papel da interação humano-computador na inclusão digital. Transinformação.* Campinas, v. 15, p. 75-89, Dec. 2003.

[7] CASAS, D. D.; MERINO, E. A. D. Gestão de design e design thinking: uma relação possível. *Revista LOGO*, v. 2, 2011.

[8] CERVO, A. L.; BERVIAN, P.; SILVA, R. *Metodologia científica.* São Paulo, Prentice Hall Brasil, 2006.

[9] CHASANIDOU, D.; GASPARINI, A. A.; LEE, E. Design Thinking Methods and Tools for Innovation. *Design, User Experience, and Usability: Design Discourse: 4th International Conference, DUXU 2015, Held as Part of HCI International 2015, Los Angeles, CA, USA, August 2-7, 2015, Proceedings, Part I.* Cham: Springer International Publishing. p. 12–23. 2015.

[10] CRUZ, T. A. DA. *Gestão de design e desenvolvimento de jogos eletrônicos:* um estudo de caso das empresas da Grande Florianópolis. 2013.

[11] CYBIS, W. *Ergonomia e usabilidade: conhecimentos, métodos e aplicações.* 3. Ed. São Paulo: Novatec, 2015.

[12] DELL'ERA, C.; VERGANTI, R. The impact of international designers on firm innovation capability and consumer interest. *International journal of operations & production management,* v. 29, n. 9, p. 870–893. 2009.

[13] DETERDING, S. Gamification: designing for motivation. *Interactions,* v. 19, n. 4, p. 14–17, 2012.

[14] DIAS, C. *Usabilidade na web: criando portais mais acessíveis*. 2. ed. Rio de Janeiro: Alta Books, 2007.

[15] EBENREUTER, N. The dynamics of design. *Kybernetes*, v. 36, n. 9/10, p. 1318–1328, 2007.

[16] FABRI, M. Thinking with a New Purpose: Lessons Learned from Teaching Design Thinking Skills to Creative Technology Students. *Design, user experience, and usability: design discourse: 4th international conference, DUXU 2015, held as part of HCI international 2015, Los Angeles, ca, USA, august 2-7, 2015, proceedings, part I.* Cham: Springer International Publishing. p. 32–43. 2015.

[17] FERNANDES, J. H. C. Qual a prática do desenvolvimento de software? *Ciência e Cultura da SBPC,* São Paulo, v. 55, n. 2. abr./jun. 2003

[18] FONSECA, Y. *Design thinking – pesquisa desk.* 2016. Disponível em: <http://blog.dtidigital.com.br/pesquisa-desk-design-tinking/>. Acesso em: 12/9/2017.

[19] FOX, R. The etymology of user experience. *Digital library perspectives*, v. 33, n. 2, p. 82–87, 2017.

[20] FROSI, F. O.; WOLFF, F.; STEFFEN, C. Gestão de game design: diretrizes para a relação entre equipes no processo de design. *Blucher design proceedings,* v. 2, n. 9, p. 1244–1253, 2016.

[21] GAIA. *GAIA soluções tecnológicas.* 2018. Disponível em: <http://gaia.uel.br/>. Acesso em: 10 jan. 2018.

[22] GARCÍA, F.; PEDREIRA, O.; PIATTINI, M.; CERDEIRA-PENA, A.; PENABAD, M. A framework for gamification in software engineering. *Journal of systems and software,* v. 132, p. 21–40, 2017.

[23] GORDON, S. R.; GORDON, J. R. *Sistemas de informação: uma abordagem gerencial.* 3ª ed. LTC, 2006.

[24] HAMDAN, S.; ALRAMOUNI, S. A Quality Framework for Software Continuous Integration. *Procedia manufacturing,* v. 3, p. 2019–2025, 2015.

[25] HODA, R.; SALLEH, N.; GRUNDY, J.; TEE, H. M. Systematic literature reviews in agile software development: A tertiary study. *Information and software technology,* v. 85, p. 60–70, 2017.

[26] IBM, K. C. *Design management capabilities*. 2017. Disponível em: <https://www.ibm.com/support/knowledgecenter/SSB2MU_8.1.0/com.ibm.xtools.rcam_prodover.doc/topics/c_dm_collab_ov.html>. Acesso em: 25 out. 2017.

[27] INCREMENTAL. *Innovation.* 2015. Disponível em: <http://www.incrementalinnovation.com/incremental-innovation/incremental-innovation-vs-radical-innovation> Acesso em: 19 set. 2017

[28] ITERAR, C. *Conceitos: iteração.* 2001. Disponível em: <http://www.funpar.ufpr.br:8080/rup/process/workflow/manageme/co_phase.htm>. Acesso em: 17 out. 2017.

[29] KHOSROWSHAHI, F. Enhanced project brief: structured approach to client-designer interface. *Engineering, construction and architectural management,* v. 22, n. 5, p. 474–492, 2015.

[30] KNUDTSON, E**.** *Concept mapping for designers of the future.* , 2016. Disponível em: <https://www.cooper.com/journal/2016/8/concept-mapping-for-designers-of-the-future>. Acesso em: 15 set. 2017.

[31] KUMAR, V. *A process for practicing design innovation.* Journal of Business Strategy, v. 30, n. 2/3, p. 91–100. 2009

[32] LAPLANTE, P. A. *What every engineer should know about software engineering.* Boca Raton: Taylor & Francis, 2007.

[33] LEIFER, R. *Radical innovation: how mature companies can outsmart upstarts.* Harvard Business Press, 2000.

[34] LINDBERG, T.; NOWESKI, C.; MEINEL, C. Evolving discourses on design thinking: how design cognition inspires meta-disciplinary creative collaboration. *Technoetic Arts*, v. 8, p. 31-37. 2010

[35] LOEWE, P.; DOMINIQUI, J. *Overcoming the barriers to effective innovation.* Strategy & Leadership, v. 34, n. 1, p. 24–31. 2006.

[36] MANFREDO, M. T. *Da hiperespecialização à integração de saberes.* 2012. Disponível em: <http://www.dicyt.com/viewNews.php?newsId=25125>. Acesso em: 26 set. 2017.

[37] MATERIAL, D. *Material design.* 2018. Disponível em: <https://material.io/guidelines/>. Acesso em: 20 jan. 2018.

[38] MCCAFFERY, F.; COLEMAN, G. Lightweight SPI assessments: what is the real cost? *Software process: improvement and practice*, v. 14, n. 5, p. 271–278, 2009.

[39] MCMANUS, J.; WOOD-HARPER, T. Software engineering: a quality management perspective. *The TQM Magazine*, v. 19, n. 4, p. 315–327, 2007.

[40] MEDEIROS, H. *Introdução ao Modelo Cascata.* 2017. Disponível em: <https://www.devmedia.com.br/introducao-ao-modelo-cascata/29843>. Acesso em: 28 set. 2017.

[41] MICHAELIS. Dicionário. *Michaelis*. Disponível em: <http://michaelis.uol.com.br/>.

[42] NAKAGAWA, E. Y. *Modelos de processo de software*. 2016. Disponível em: <https://edisciplinas.usp.br/pluginfile.php/839466/mod_resource/content/1/Aula02_ModelosProcessos.pdf>. Acesso em: 6 dez. 2017.

[43] NICHOLSON, S. A user-centered theoretical framework for meaningful gamification. *Games learning society,* v. 8, n. 1, p. 223–230, 2012.

[44] NIELSEN, J. *Usability engineering*. Cambridge, MA: Academic Press, 1993.

[45] NIELSEN, J.; LORANGER, H. *Usabilidade na web: projetando websites com qualidade*. Rio de Janeiro: Elsevier, 2007.

[46] NORMAN, C.; JERRARD, R. Design managers, their organizations and work-based learning. *Higher education, skills and work-based learning*, v. 5, n. 3, p. 271–284, 2015.

[47] NSU. *From customer focus to customer obsession*. , 2018. Disponível em: <https://secure.business.nova.edu/marketing-blog/index.cfm/postedby/admin>. Acesso em: 8 jan. 2018.

[48] OGAWA, F. S. *Recomendações para o desenvolvimento de hipermídias acessíveis a crianças surdas*. 182 fls. Trabalho de Conclusão de Curso (Graduação em Design Gráfico) - Universidade Estadual de Londrina, Londrina. 2010.

[49] PENG, Q.; MATTERNS, J. B. Enhancing User Experience Design with an Integrated Storytelling Method. *Design, User Experience, and Usability: Design Thinking and Methods: 5th International Conference, DUXU 2016, Held as Part of HCI International 2016, Toronto, Canada, July 17–22, 2016, Proceedings, Part I*. Cham: Springer International Publishing. p. 114–123. 2016

[50] PRESSMAN, R. S. *Software engineering: a practitioner's approach*. 5th ed. Boston, Mass: McGraw Hill, 2000.

[51] PRINCIPLES. *Manifesto for agile software Development*. 2001. Disponível em: http://agilemanifesto.org/. Acesso em: 19 nov. 2017.

[52] ROLIM, C. O. *Introdução ao design thinking*. 2017. Disponível em: <http://www.san.uri.br/~ober/arquivos/disciplinas/mestrado_adm/design_thinking.pdf>. Acesso em: 14 dez. 2017.

[53] ROSE, D.; MEYER, A. *Teaching every student in the digital age*: Universal Design for Learning. Alexandria, VA: ASCD. 2012.

[54] SANCHEZ-GORDON, M.-L.; DE AMESCUA, A.; O'CONNOR, R. V.; LARRUCEA, X. A standard-based framework to integrate software work in small settings. *Computer Standards & Interfaces*, v. 54, p. 162–175, 2016.

[55] SATO, S. Beyond good: great innovations through design. *Journal of Business Strategy*, v. 30, n. 2/3, p. 40–49, 2009.

[56] SCHIMIGUEL, J. *Agile Development: XP e Scrum em uma Abordagem Comparativa*. Disponível em: <https://www.devmedia.com.br/agile-development-xp-e-scrum-em-uma-abordagem-comparativa/30808>. Acesso em: 26 set. 2017.

[57] SCRUM. *O Ciclo de Vida Scrum*. 2018. Disponível em: <http://engeprojnews.blogspot.com.br/2013/04/o-ciclo-de-vida-scrum.html>. Acesso em: 06 jan. 2018.

[58] SHMUELI, O.; RONEN, B. Excessive software development: Practices and penalties. *International Journal of Project Management*, v. 35, n. 1, p. 13–27, 2017.

[59] SHNEIDERMAN, B.; PLAISANT, C.; COHEN, M.; JACOBS, S.; ELMQVIST, N., *Designing the User Interface*: Strategies for Effective Human-Computer Interaction: Sixth Edition, Pearson. 2016.

[60] SMITE, D.; MOE, N. B.; SABLIS, A.; WOHLIN, C. Software teams and their knowledge networks in large-scale software development. *Information and Software Technology*, v. 86, p. 71–86. 2017

[61] SOMMERVILLE, Ian. *Engenharia de Software*, Person, São Paulo, 2010.

[62] STEINMACHER, I.; GRACIOTTO SILVA, M. A.; GEROSA, M. A.; REDMILES, D. F. A systematic literature review on the barriers faced by newcomers to open source software projects. *Information and Software Technology*, v. 59, p. 67–85, 2015.

[63] TIDD, Joe; BESSANT, John; PAVITT, Keith. *Gestão da Inovação*. 3. ed. Porto Alegre: Bookman, 2008.

[64] USABILIDADE. *Usabilidade e diversão em jogos digitais.* Disponível em:< http://gamereporter.uol.com.br/usabilidade-e-diversao-em-jogos-digitais/>. Acesso em: 28 abril 2018.

[65] VIANNA, M.; VIANNA, Y.; ADLER, I. K.; LUCENA, B.; RUSSO, B. *Design thinking: inovação em negócios*. Design Thinking, 2011.

[66] WARD, A.; RUNCIE, E.; MORRIS, L. *Embedding innovation:* design thinking for small enterprises. Journal of Business Strategy, v. 30, n. 2/3, p. 78–84. 2009.

[67] WAZLAWICK, R. S. *Engenharia de software: conceitos e práticas.* Rio de Janeiro: Elsevier. 2013.

[68] WOHLIN, C.; ŠMITE, D.; MOE, N. B. A general theory of software engineering: Balancing human, social and organizational capitals. *Journal of Systems and Software,* v. 109, p. 229–242, 2015.

[69] XIMENES, B. H.; ALVES, I. N.; ARAÚJO, C. C. Software Project Management Combining Agile, Lean Startup and Design Thinking. *Design, User Experience, and Usability: Design Discourse: 4th International Conference, DUXU 2015, Held as Part of HCI International 2015, Los Angeles, CA, USA, August 2-7, Proceedings, Part I.* Cham: Springer International Publishing, 2015. p. 356–367. 2015.

[70] YILMAZ, M.; O'CONNOR, R. V.; COLOMO-PALACIOS, R.; CLARKE, P. An examination of personality traits and how they impact on software development teams. *Information and Software Technology*, v. 86, p. 101–122, 2017.

[71] ZICHERMANN, G.; CUNNINGHAM, C. *Gamification by design: Implementing game mechanics in web and mobile apps*. O'Reilly Media, Inc., 2011.

[72] ZUEHLKE, D.; THIELS, N. Useware engineering: a methodology for the development of user-friendly interfaces. *Library Hi Tech*, v. 26, n. 1, p. 126–140, 7 mar. 2008.

APÊNDICES

APÊNDICE A – QUESTIONÁRIO

1. Levando-se em conta um projeto divido em 5 grandes fases: Levantamento de Requisitos, Análise e Design, Prototipação, Implementação e Testes, baseado na sua experiência, em quais dessas fases ocorre os maiores deslizes que levam ao fracasso (ou perda de qualidade) de um projeto de software.

- Levantamento de Requisitos;
- Análise e Design;
- Prototipação;
- Implementação;
- Testes;
- Justifique.

2. Em sua opinião e experiência, qual principal motivo para que projetos de desenvolvimento de software falhem em cumprir alguns ou todos os requisitos de projeto.

- Falta de compreensão das reais necessidades do projeto;
- Erro de estimativa do esforço e/ou cronograma necessários para o desenvolvimento;
- Falta de/Desalinhamento de habilidade técnica por parte da equipe.
- Falta de comunicação eficaz do progresso individual;
- Falta de Ambiente virtual para exposição de ideias;
- Inexistência de metodologia bem estrutura para gerenciamento do projeto;
- Gerenciamento fraco ou inexistente;
- Falta de conhecimento/Resistência as práticas de gerenciamento de projeto por parte da equipe;
- Incompatibilidade entre membros da equipe;
- Outro.

3. Abaixo atribua uma nota de 0 a 5, menor ao maior impacto, quanto aos requisitos expostos que você acredita que tenha afetado a qualidade de um software e seu processo de construção no qual você tenha participado.

Quanto à equipe.

- Falta de interação/comunicação direta e constante com os membros da equipe;
- Demora em respostas e feedbacks;
- Falta de expertise técnica por parte da equipe;
- Falta de conhecimento sobre gerenciamento de projeto e práticas metodológicas;
- Desalinhamento de nível de habilidade entre membros da equipe;
- Dificuldade em localizar tarefas apropriadas para se iniciar um projeto;
- Alta gerência fraca e/ou com pouco controle sob os demais membros da equipe;
- Incompatibilidade entre membros da equipe.

Em termo de gerenciamento e metodologia de processo.

- Falta de acesso/explicação por parte da gerência quanto ao framework de trabalho que deverá ser seguido;
- Muita documentação;
- Falta de clareza na troca de feedback entre membros;
- Falta de um ambiente virtual ou físico para a produção da documentação do progresso (individual e coletivo) do desenvolvimento do projeto. (Um lugar onde todos os membros da equipe possam expor visualmente suas ideias/opiniões e progressos em relação ao projeto, e que possam analisar constantemente a evolução de outros).

Quanto à interação entre designer de interface e programador.

- Falta de conhecimento básico acerca da área do colega. (Ex: Um designer que não possui conhecimento mínimo de como o programador trabalha ou vice-versa);
- Falta de conhecimento de termos técnicos;
- Falta de domínio técnico por parte do programador de programas de design como Photoshop, Illustrator;
- Falta de domínio de linguagem descritiva por parte do designer na hora de passar a documentação para construção da interface em código. O designer utiliza termos alheios a área de computação que não se aplicam ao que realmente precisa (Ex: utiliza sistema de códigos de cor que não está em código Hex, utilizar medidas como cm/mm/pts ao invés de pixel ou densidade de pixels).

4. Se você já trabalhou em uma equipe onde era o único de sua área, quando o foco de uma reunião era voltado a questões alheias a sua área de expertise, havia esforço para se inteirar sobre esforço, ou você acredita ser desnecessário participar de discussões que vão além do seu domínio.

5. Se você é da área TI, já teve dificuldades em trabalhar com alguém da área de Design. Se sim, o que você acredita ser o desafio de alinhas o desenvolvimento interface com as práticas da computação?

6. Se você é da área de Design, já teve dificuldades em trabalhar com alguém da Computação em um projeto de desenvolvimento de software. Se sim, o que você acredita ser o desafio de alinhar o desenvolvimento interface com as práticas da computação?

APÊNDICE B – AVALIAÇÃO FINAL

O questionário abaixo deve ser respondido após a leitura da descrição do framework proposto, contido neste documento. Leia as afirmações abaixo e marque com um X, se concorda ou discorda do que foi proposto para o framework GAIA.

1 Discordo Totalmente e 5 Concordo Totalmente.

1. A Inserção do design e suas ferramentas em fases iniciais propicia resultados mais alinhados de interface e programação, evitando também retrabalhos.

1 2 3 4 5

2. A ênfase maior no levantamento de requisitos em prol de um planejamento adequado ao projeto e maior domínio das necessidades do mesmo, propiciam consolidação de fases e resultados mais consistentes com as expectativas do usuário/cliente.

1 2 3 4 5

3. A divisão em iterações (porções de trabalho) menores em um laboratório de TI, que conta com equipes pequenas e inexperientes, é a forma mais recomendada de abordar o desenvolvimento de software, pois o acumulo de erros é drasticamente reduzido.

1 2 3 4 5

4. A inserção de testes com mais frequência (recomenda-se um ao fim de cada iteração) elimina retrabalhos e a colabora com a consolidação de fases e um software mais refinado a cada etapa.

1 2 3 4 5

5. O uso da dinâmica SCRUM de comunicação (sempre comunicar o progresso feito em uma atividade e reuniões periódicas cara a cara ao fim de cada iteração), e o quadro de atividades, ajudam a reduzir o problema de comunicação que equipes de software normalmente enfrentam.

1 2 3 4 5

6. A participação do profissional de TI nas ferramentas e abordagens de design

thinking, propiciam insights e maior interesse pela área de criação por parte do programador, e desenvolvimento mais alinhado entre interface e programação.

1 2 3 4 5

7. A ferramenta de Briefing entre equipe (como foi proposto) onde deve haver troca de conhecimento entre programador e designer (acerca do método e experiência de trabalho de cada um) propicia o nivelamento da equipe e definição de fases mais condizente com a habilidade geral da equipe.

1 2 3 4 5

8. A inserção da jornada do designer no framework reduz a confusão de qual a importância e o papel deste profissional e de sua disciplina dentro do projeto, ou seja, com sua participação abrangente desde as fases iniciais, passando por levantamento de requisitos, implementação, testes e análises, elimina a ideia do uso do designer como mera ferramenta estética e o consolida como fator essencial para bons resultados em todas as etapas de criação.

1 2 3 4 5

9. De forma geral, qual nota você daria para o framework proposto, em sua tentativa de diminuir a lacuna que existe entre as duas áreas de desenvolvimento de software.

1 2 3 4 5

TRABALHOS PUBLICADOS PELO AUTOR

Trabalhos publicados pelo autor durante o programa.

1. BERGAMIM, A. F., BARROS, R. M., BARROS, V.T., SILVA, M. S. *A Spiral-Based Approach to the Interface Development of Assisted-Interaction Software for Deaf Patients in Dental Treatment*, CONTECSI, 04/2018 (Qualis B4 2018).