

De Volta ao Ponto: Uma abordagem prática para re-alinhamento de backlogs

TRABALHO DE CONCLUSÃO DE CURSO

Diego dos Santos Centeno

Pós-Graduação em Tecnologias Aplicadas a Sistemas de Informação com Métodos Ágeis
Centro Universitário Ritter Dos Reis - UNIRITTER
centeno@diegocenteno.com

Émerson Barbiero Hernandez

Professor Orientador
emerson.hernandez@gmail.com

Abstract

Uma Concepção (em inglês, Inception) é uma dinâmica com foco na descoberta do produto, aplicada a projetos de software. Esta atividade visa promover um maior alinhamento entre o time de desenvolvimento e o cliente em relação ao conceito do produto e as expectativas gerais. O objetivo de uma Concepção é preparar o time envolvido para um desenvolvimento inteligente e ágil do produto proposto, reduzindo o risco de conflitos de expectativas e retrabalho. O resultado desses experimentos é ter uma equipe alinhada quanto às necessidades e prioridades que o desenvolvimento do produto requer no curto, médio e longo prazo. Porém o mundo se move rápido e as necessidades de negócio mudam o tempo todo, então é preciso constantemente verificar se o planejamento existente atende as expectativas de negócio ou se precisamos modificar o planejado para se adaptar as novas necessidades do mercado. Neste trabalho vamos apresentar uma abordagem prática de Concepção Enxuta (em inglês, Lean Inception) que foca no alinhamento e validação de um backlog existente através de um conjunto de dinâmicas e metodologias já existentes, porém já testadas e validadas em um ambiente real.

Palavras-chave: Metodologia ágil, Concepção Enxuta, Lean.

1 Introdução

As metodologias ágeis têm sido cada vez mais apontadas como a melhor alternativa para o desenvolvimento de software. O termo 'ágil' refere-se a uma filosofia de desenvolvimento de software. Elas propõem uma nova abordagem de desenvolvimento, eliminando gastos com documentação excessiva e burocrática, enfatizando a interação entre pessoas e nas atividades que efetivamente trazem valor e produzem software com qualidade [1]. Podemos também dizer que a aplicação das metodologias ágeis no desenvolvimento de software, ou simplesmente desenvolvimento ágil de software, é o desenvolvimento iterativo e incremental onde as hipóteses (ou requisitos) são testadas e implementadas por colaboração entre pequenos times auto-organizados e multifuncionais. [2] Debaixo deste amplo guarda-chuva estão abordagens mais específicas como Extreme Programming (XP), Scrum e Desenvolvimento Enxuto (Lean). Cada uma dessas abordagens em particular possui suas próprias ideias, comunidades e líderes. [3]

Dentro deste novo paradigma uma prática muito emergente é criar colaborativamente o entendimento sobre o produto a ser desenvolvido antes de sair simplesmente programando. Diferentes abordagens têm sido propostas para que os times descubram e entendam coletivamente o escopo se ser desenvolvido, dentre elas a mais comum é a Concepção (em inglês, Inception).

Diante das várias possibilidades de formatos de se fazer uma concepção, uma que vem se destacando em número de adeptos e projetos é a Concepção Enxuta (Lean Inception), também conhecida como Direto ao Ponto. [4] Ela consiste em um conjunto de atividades com a finalidade de clarear e alinhar o objetivo do produto, aprofundar para quem é este produto e como esse usuário interage com ele, assim pensando em suas funcionalidades e após faz um sequenciamento dessas funcionalidades para determinar qual o mínimo produto viável.

No entanto, devido a complexidade de um projeto e as todas as tarefas do dia-a-dia, perdemos de vista aquilo que queremos alcançar. Dada a natureza do trabalho colaborativo, juntamente com os pontos citados anteriormente, é necessário de tempos em tempos realinhar o escopo das funcionalidades inicialmente pensadas na Inception. Com isto precisamos voltar ao planejamento inicial, ou seja, precisamos voltar ao Direto ao Ponto. Este artigo apresenta uma abordagem prática do uso de dinâmicas para re-validar e re-alinhar as funcionalidades e MVPs que surgiram na Inception Enxuta, chamado de "De Volta ao Ponto".

2 Revisão de Literatura

2.1 Design Thinking

O *Design Thinking* é um processo colaborativo de pensamento crítico e criativo que permite organizar informações e ideias, para tomar decisões, aprimorar situações e adquirir conhecimento [5]. Sendo assim, pode-se dizer que ele é um método que trabalha essencialmente a cocriação e utiliza algumas práticas valiosas de descoberta. Desta forma, e até mesmo por possibilitar

a coleta de resultados imediatos a partir de ações simples, acaba por auxiliar também no acultramento da empresa. O *Design Thinking* é um método bastante eficaz e presente nas organizações que buscam realmente entender, avaliar e implementar o que os seus clientes e/ou *stakeholders* pensam sobre seus produtos e serviços. A sua utilização abrange diferentes ramos da indústria, podendo ser considerada também uma ferramenta estratégica que incentiva a criação conjunta a partir de uma visão por processos. E nesta perspectiva, o processo de Design Thinking é definido, conforme as seguintes fases [5]:

- **Imersão** (entendimento) - envolve a contextualização do projeto;
- **Ideação** (criação) - tem o intuito de gerar ideias inovadoras para o tema de projeto;
- **Prototipação** (teste) - tem como função validar hipóteses a partir da formulação de questões, criação de protótipos, testes, avaliações e conclusão;
- **Desenvolvimento** (aplicação) - o próprio uso do conhecimento adquirido através do processo criativo.

2.2 Desenvolvimento Enxuto

Desenvolvimento Enxuto (em inglês Lean Development) é inspirado na abordagem do mesmo nome aplicado aos sistemas de produção, que foi iniciado por Taiichi Ohno na Toyota e é também conhecido como “Sistema Toyota de Produção” [6]. Essa filosofia de gerenciamento de processos transformou completamente a abordagem da fabricação de automóveis, que evoluiu para um modelo mais responsável, muito mais sustentável e, ao mesmo tempo, mais eficiente.

O desenvolvimento enxuto originou-se de um livro popular de Tom e Mary Poppendieck onde foram incorporados os princípios da fabricação enxuta para o desenvolvimento de software. [7] Os seus benefícios contribuíram para a disseminação rápida desta metodologia, que agora é generalizada entre a comunidade global de métodos ágeis.

Há uma série de aspectos que configuraram este modelo de desenvolvimento. Sua aplicação é uma condição necessária para a existência de ajuste e, portanto, são conhecidos como os 7 princípios fundamentais do desenvolvimento de software enxuto:

1. **Eliminar o desperdício:** Maximizar o trabalho a não ser feito, seja evitando retrabalhos, processos e funcionalidades extras, defeitos, estoque de tarefas, atrasos e esperas;
2. **Amplificar conhecimento:** Priorizar a comunicação e o *feedback* contínuos entre equipes e usuários durante o processo de desenvolvimento de software;
3. **Adiar decisões:** Deixar as decisões e comprometimentos para o último momento responsável, permitindo coletar informações e ter experiências para fortalecer a tomada de decisão;
4. **Entregas rápidas:** Maximizar o retorno do investimento (*ROI*) do projeto, entregando software de valor de forma rápida e contínua;
5. **Fortalecer o time:** Empoderar o time e criar um ambiente onde a equipe trabalhe de forma auto-organizada e auto-dirigida, evitando micro-gerenciamento;
6. **Construir qualidade:** Garantir qualidade no desenvolvimento do software utilizando técnicas como TDD, refatoração e integração continua;
7. **Otimizar o todo:** Entender que o software concluído é muito mais que a soma das partes entregues e verificar como ele está alinhado com os objetivos da empresa.

2.3 Concepção Enxuta

A Concepção Enxuta, como o próprio nome sugere, é uma abordagem mais enxuta de concepção. Esse método consiste em um laboratório colaborativo, onde em uma semana de trabalho, o plano é compreender o objetivo do projeto, identificar os principais usuários que terão contato com a solução e o escopo funcional de alto nível, ter insumos para definição da estratégia de lançamento incremental dos *MVPs* (em português, Produto Mínimo Viável). A metodologia de Concepção Enxuta é uma técnica para compreender e planejar as entregas incrementais dos *MVPs*. A técnica organiza as ideias e funcionalidades em um modelo que busca compreender o objetivo principal do produto, considerando as jornadas dos usuários e os incrementos de entrega.

Com atividades eficazes e rápidas, a técnica desafia as considerações tradicionais de projeto, como análise de requisitos, estimativas, escopo, capacidade e planejamento detalhado. Ao longo da execução de um Concepção Enxuta devemos alcançar as seguintes etapas:

- Descrever a visão do produto;
- Priorizar os objetivos de negócio;
- Descrever os usuários, seus perfis e necessidades que impulsionam a funcionalidade do produto;
- Entender o escopo funcional de alto nível;

- Detalhar percepções de risco, esforço e valor de negócio por área funcional;
- Descrever as principais jornadas de usuários;
- Definir um roteiro de lançamento incremental, mostrando claramente os incrementos de *MVP*;
- Estimar esforço por amostragem;
- Calcular custos e especificar datas e cronograma de entrega.

3 Abordagem prática

A abordagem apresentada aqui é, assim como no modelo de Concepção Enxuta, um *workshop* colaborativo. O foco deste evento é realinhar os membros da equipe sobre o objetivo do projeto, sobre o escopo que ainda precisa ser feito e mais do que tudo sobre o caminho a seguir. Este *workshop* consiste basicamente em executar uma sequência de atividades e dinâmicas, baseadas em *design thinking*, para agrupar, organizar, detalhar e priorizar o trabalho e o caminho a ser seguido. Após realizarmos a concepção, saímos com uma lista de funcionalidades a serem executadas. O nome dado a esse conjunto de funcionalidades é *backlog*.

Entretanto, diversos motivos inerentes da natureza do trabalho iterativo e incremental nos levam a perder de vista aquilo que realmente queremos como objetivo. Objetivo esse que foi desmembrado em diversas funcionalidades dentro do *backlog*. Alguns motivos podem ser mais básicos, como por exemplo o alto nível de envolvimento da equipe nas tarefas do dia-a-dia, ou até mesmo motivos mais complexos, como por exemplo a rotatividade de membros da equipe.

Por esta razão, é muito importante re-alinhar, como equipe, se o *backlog* gerado anteriormente está no rumo correto para que o objetivo seja atingido. Fazendo isto podemos, no mínimo, ter alguns ganhos para o produto/projeto ou até mesmo para a organização:

- **Validar o que já foi feito:** Validar se as hipóteses imaginadas e as funcionalidades já desenvolvidas efetivamente atenderam as necessidades do produto/projeto;
- **Revalidar o caminho a seguir:** Projetos assumem um caminho a seguir, mas, na prática o caminho é mais distorcido do que planejamos; esta dinâmica ajuda a revalidar o caminho, subindo e baixando de nível, considerando o conhecimento atual e o objetivo a alcançar;
- **Gerar entendimento compartilhado:** É sempre importante que o objetivo e o contexto do que precisa ser feito seja de conhecimento de todos os membros da equipe;
- **Reducir ramp-up do onboarding:** este realinhamento do *backlog* pode ser aproveitado para gerar contexto para quem está entrando no time, diminuindo assim a curva de aprendizado para quem está entrando;
- **Reducir riscos:** Quanto mais vezes checamos se estamos alinhados e caminhando em direção ao objetivo, menos riscos corremos de não atingir ou até mesmo fugir do objetivo.

3.1 Definindo os objetivos

Primeiro é necessário subir de nível, deixar o *backlog* um pouco de lado, e entender os objetivos do projeto ou da equipe. Para isto, utilizamos a técnica do 'Diamante duplo' [8], em sua primeira parte: Descobrir e Definir. O primeiro diamante, conforme apresentado na figura 1, sugere que inicialmente deve-se gerar opções a fim de encontrar novas possibilidades e a partir dele, gerar opções.

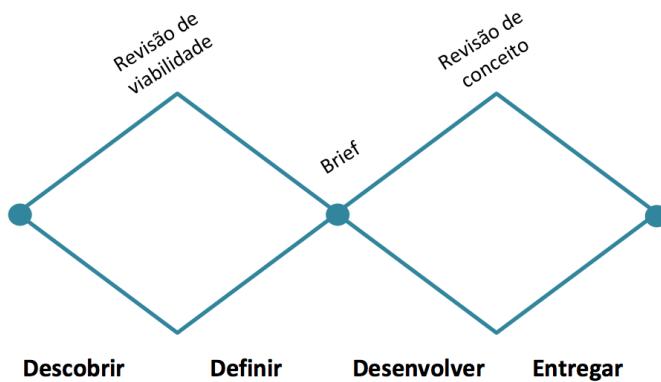


Figura 1: Diamante Duplo

Primeiramente, abrimos espaço para que uma série de ideias sejam descobertas e compartilhadas entre os membros da equipe. Normalmente fazemos isso com o formato de um *brainstorm*. Esse é o momento de divergir e descobrir. O objetivo não é criar uma lógica racional ou traçar um caminho de solução para o problema, procura-se gerar o maior número de alternativas e contextos possíveis e a partir disso, decidir qual são as melhores opções. Feito isto, é momento de convergir e definir. Nesta segunda etapa deve-se discutir com a equipe os objetivos levantados a fim de identificar quais são potencialmente semelhantes e então agrupá-los. Devemos sair desta etapa com no máximo 5 objetivos por três simples motivos:

- **Foco:** Ter muitos objetivos pode fazer com desviemos o foco ao longo do caminho, com isso podemos acabar não atingindo nenhum deles.
- **Validação:** Manter o foco em poucos objetivos vai fazer com que o time seja capaz de rodar periodicamente esta dinâmica para revalidar os mesmos.

3.2 Aplicando OKRs

De posse dos objetivos identificados na atividade anterior (3.1), agora é a vez de utilizarmos a metodologia de *OKRs* para estabelecer o caminho para atingirmos os mesmos. A metodologia OKR (Objectives and Key Results) nasceu dentro da Intel, no Vale do Silício, com a missão de estabelecer objetivos e resultados-chave para mensurar o quanto próximo se está do objetivo. Essa metodologia vem sendo adotada desde a década de 90 em vários empreendimentos, independente de tamanho, desde grandes corporações até startups. [9] Na metodologia OKR, primeiro você cria um objetivo e, em seguida, configura um conjunto de resultados chave: [10]:

1. **Objetivos (Objective):** É uma descrição qualitativa do que deseja alcançar. Os objetivos devem ser curtos, inspiradores e envolventes. Um Objetivo deve motivar e desafiar a equipe.
2. **Resultados chave (Key Results):** São um conjunto de métricas que medem o seu progresso e o impacto direto no atingimento do objetivo, caso seja realizada com sucesso. Para cada Objetivo, você deve ter um conjunto de 2 a 5 resultados chave.

John Doerr, idealizador da metodologia na Intel e também responsável por sua implementação no Google, criou um modelo bastante simples de aplicação de *OKRs*: [11]

Eu vou objetivo mensurado por lista de resultados chave

Utilizando o modelo acima, devemos criar de 3 a 5 resultados chave para cada objetivo. O conjunto destes resultados chave é, para nós, o balizador para saber se atingimos o objetivo ou não. Ao final desta atividade, devemos ter uma lista que se assemelha a figura 2.

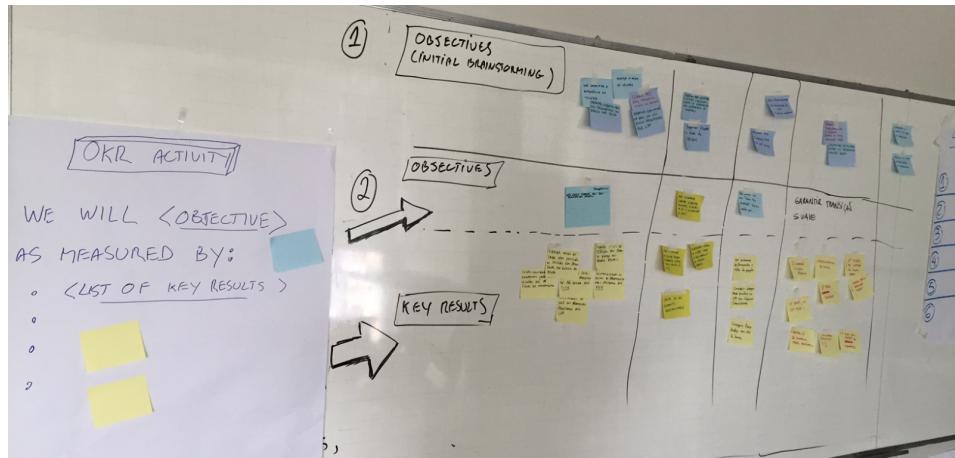


Figura 2: Exemplo de como aplicar o modelo de *OKRs*

3.3 Backlog

A partir desta etapa, vamos passar a trabalhar exclusivamente com o nosso *backlog*. A ideia é entrar nessa atividade com o *backlog* atual e, ao final da abordagem, sair com um novo *backlog* (seja por incluir novas funcionalidades, cortar algumas que não fazem mais sentido ou até mesmo pelo simples fato de re-priorizar o mesmo).

A primeira coisa a ser feita é listar todo o *backlog* atual e garantir que todos os itens desta lista estão no mesmo nível de abstração. Durante as dinâmicas que vamos aplicar estaremos trabalhando sempre a nível de funcionalidades, mas podem também ser trabalhado em qualquer nível abstração, como estórias, épicos ou até mesmo tarefas. Neste momento não há necessidade alguma de vincular os itens do *backlog* com a prática de *OKR* que fizemos anteriormente, afinal de contas, indiretamente, vamos estar olhando para as funcionalidades já com a visão que definimos na dinâmica anterior.

3.4 Método da cebola

De posse da lista de funcionalidades (histórias ou épicos), é o momento de identificarmos e validarmos o quanto destas funcionalidades já foram implementadas. Existem diversas formas de fazer isto, porém aqui vamos utilizar uma dinâmica criada pelo Lourenço Soares. [12]

Inicialmente o método da cebola foi criado e utilizado para escrever as histórias de uma funcionalidade e ou épico. Ele consiste idealmente em ir descrevendo as histórias em camadas, ou seja, coloca-se a tarefa mais básica no núcleo da cebola, após se cria uma camada em volta deste centro e se coloca as tarefas um pouco mais complexas. Repete-se isto até que todas as tarefas da história sejam mapeadas, como pode ser visto na figura 3.

Como já temos as nossas histórias prontas e o que buscamos é identificar quanto destas funcionalidades já foram implementadas, iremos utilizar o método da cebola de uma forma um pouco distinta. Iremos utilizar conceito de camadas do método da cebola para validar o quanto já avançamos com a funcionalidade e, para isto, vamos utilizar apenas três camadas (vide figura 3). A ideia aqui é que, para cada história, possamos saber o quanto já foi concluído e assim atribuirmos a história uma *tag*:

- **R1:** significa que tudo ou muito ainda precisa ser feito
- **R2:** significa que já avançamos bastante, porém ainda dá algumas coisas para serem feitas
- **R3:** significa que já estamos muito próximos de terminar

É comum que neste momento a equipe se dê por conta que determinada funcionalidade já foi implementada em uma outra história ou até mesmo que essa funcionalidade não faz mais sentido para o projeto, produto ou para a própria equipe.

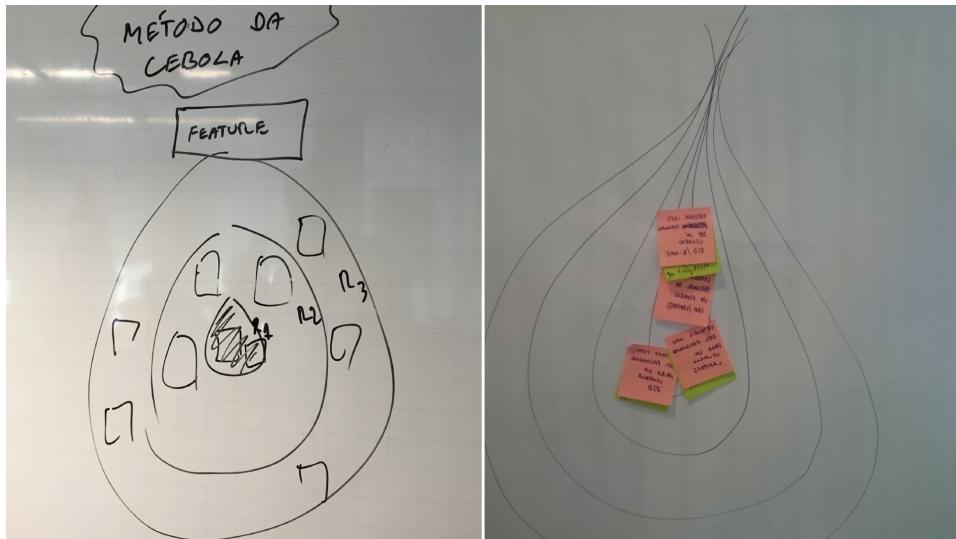


Figura 3: Exemplo do método da cebola / Exemplo de aplicação do método da cebola

3.5 Canvas de esforço versus valor de negócio

Uma vez que já conhecemos o nosso *backlog* e identificamos o quanto já avançamos com as funcionalidades, agora é hora de identificar o nível de esforço e o valor de negócios associado a cada funcionalidade. Faremos isso seguindo os passos sugeridos pelo Paulo Caroli no livro *Direto Ao Ponto* [4]:

1. Vamos criar um *canvas* (gráfico) com dois eixos, onde o eixo X vai representar o esforço (a dificuldade e o trabalho que será necessário para completar tal funcionalidade) e o eixo Y vai representar o valor para o negócio desta funcionalidade.

Também iremos criar uma escala para um dos eixos (conforme figura 4), no eixo de esforço vamos adotar: *E*, *EE* e *EEE* (onde *E* representa um baixo esforço e *EEE* representa um alto esforço), no eixo de valor de negócios vamos adotar: *\$*, *\$\$* e *\$\$\$* (onde *\$* representa um baixo esforço e *\$\$\$* representa um alto esforço).

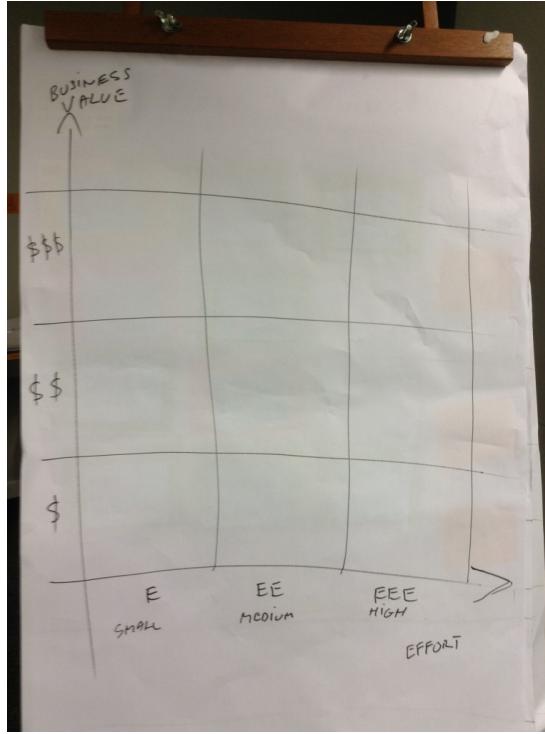


Figura 4: *Canvas* de esforço versus valor de negócio

2. Agora vamos pegar uma funcionalidade do nosso *backlog* e vamos posicionar no *canvas* de acordo com o entendimento que o time tem sobre o esforço e o valor da mesma. Neste momento é importante levar em consideração a estimativa que fizemos no através do método da cebola (3.4), pois ela pode impactar diretamente na estimativa de esforço.
Neste momento é importante que toda equipe partilhe da mesma opinião. Se algum membro do time não concordar com a escala aplicada, o time deve discutir os requisitos e a tecnologia envolvida de forma que haja um consenso sobre a escala da funcionalidade.
3. Uma vez classificada a funcionalidade na escala de esforço e de valor, vamos marcar a mesma de acordo com escala de esforço (E, EE ou EEE) e de valor (\$, \$\$ e \$\$\$) que foi atribuída (vide figura 5).
4. Por fim, basta repetir os passos 2 e 3 até que todas as funcionalidades do *backlog* possuam um esforço e um valor atribuído.



Figura 5: Exemplo de funcionalidade com marcações de esforço e valor

3.6 Sequenciador de funcionalidades

Dado que já sabemos os nossos objetivos (levantados através dos *OKRs*) e que também já possuímos todas as funcionalidades classificadas com esforço e valor, agora é o momento de priorizarmos as mesmas. Isso vai garantir que seja possível ter uma

previsibilidade mínima além de destacar os *milestones* e/ou ter uma previsão de atingimento dos *OKRs*.

O sequenciado de funcionalidades, assim como o *canvas* de esforço e valor, é um gráfico (*canvas*). Porém ao invés de ser dividido em blocos, ele é dividido apenas em linhas, o que aqui vamos chamar de ondas (vide figura 6).

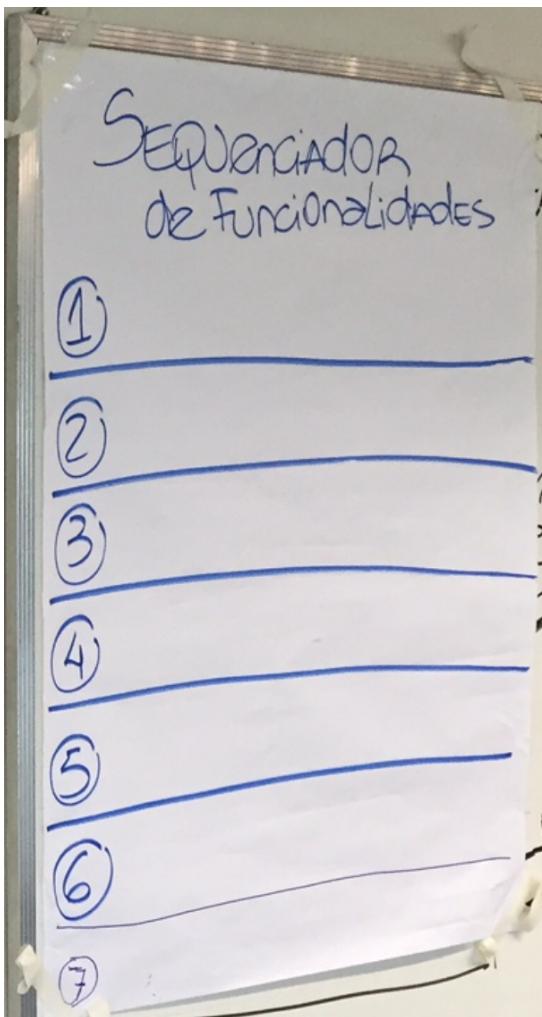


Figura 6: Exemplo de sequenciador de funcionalidades

Agora é o momento do time, em conjunto, decidir e organizar a sequencia de execução das funcionalidades. O time se organizar e distribuir as funcionalidades dentre as ondas, onde a primeira onda vai conter as funcionalidades que o time quer entregar primeiro e assim sucessivamente até a ultima onda.

Neste momento é importante levar em consideração as estimativas de esforço e valor que fizemos no através do *canvas* de esforço versus valor (3.5). Além disto é muito importante seguir algumas regras para o sequenciador seja mais efetivo:

1. Uma onda pode conter no máximo 3 funcionalidades.
2. A soma de esforço das funcionalidades não pode ultrapassar 5 Es.
3. A soma de valor de negocio das funcionalidades não pode ser menos de 4 \$s.
4. Uma onda tem de conter no mínimo 2 funcionalidades.

As regras acima visão limitar o número de funcionalidades que serão trabalhadas ao mesmo tempo em cada onda (regra 1), evitam um período de trabalho desequilibrado com muita incerteza ou muito esforço (regra 2) e garantem um foco constante na maior entrega de valor para o negócio (regra 3 e 4).

4 Considerações finais

O desenvolvimento do presente trabalho possibilitou uma análise de como as metodologias de Concepção Enxuta podem melhorar os resultados do processo de desenvolvimento de software ágil. Além disso, também permitiu evidenciar que não é necessário ficar preso as grandes metodologias e/ou processos de criação, ou seja, podemos adotar ou até mesmo criar dinâmicas que mais se adéquam as nossas necessidades.

Durante a criação deste trabalho tivemos a oportunidade de rodar a prática descrita na seção 3 e assim evidenciar de forma mais consistente os ganhos aqui propostos. Rodamos a prática em 6 projetos, dos quais 3 já haviam passado pelo método [4] e tinham sua visão de produto e seus *backlogs* muito bem alinhados, e outros 3 projetos que não fizeram nenhum tipo de Concepção e que apenas tinham como *backlog* uma lista de funcionalidades que foram criadas a partir do projeto em andamento.

O maior ganho evidenciado é a geração colaborativa de conhecimento do produto, o que reflete a curto e médio prazo na qualidade, acuracidade das entregas do produto, o que tende a impactar diretamente na diminuição dos riscos e na redução dos custos do projeto. Também foi possível observar que nos times que rodaram a prática, conseguimos atingir os seguintes resultados:

- Refinar e /ou corrigir erros de previsibilidade (estimativas) por já conhecer mais do contexto do produto/projeto;
- Simplificar coisas e processos dado o conhecimento adquirido durante o desenvolvimento prévio;
- Reconhecer que as tecnologias envolvidas evoluíram e que é possível adotar ou escolher novas alternativas;
- Criar mais foco nas funcionalidades faltantes, dado ao fato de construir uma visão mais unificada do que ainda falta;
- Guiar o desenvolvimento e as prioridades do *backlog* orientado aos objetivos do produto/projeto;
- Criar uma fator motivacional extra através do fortalecimento e alinhamento do time, uma vez que estão todos juntos fazendo a dinâmica;
- Criar uma visão unificada do time sobre os objetivos do produto/projeto, sobre tudo em times que tiveram alterações de seus membros;
- Re-priorizar funcionalidades que já não são mais tão importantes dados os objetivos do produto/projeto.

Por se tratar de uma prática de exploração mais do que uma técnica exata aliado ao número ainda pequeno de execuções não permitiu obter dados (números e/ou indicadores) mais consistentes sobre os ganhos da aplicação da metodologia. Uma pesquisa mais aprofundada para obter dados mais consistentes sobre os ganhos de outras metodologias de Concepção e Concepção Enxuta foi realizada, mas atualmente existem poucos dados acessíveis que evidenciem o ganho direto desta metodologias.

Deve-se levar em consideração que, apesar de ter muitas referencias diretas ao método *Direto Ao Ponto* [4], este trabalho aborda outros aspectos da *Concepção Enxuta*. Enquanto o *Direto Ao Ponto* foca na construção do produto e/ou projeto na sua fase inicial, passando desde a visão do produto até a construção de um *backlog* e seus MVPs, este trabalho visa atender principalmente time que já possuem um *backlog* definido e que necessitam validar se as funcionalidades planejadas ainda estão de acordo com os rumos que o negocio tomou.

Na sequência do presente trabalho surgiram alguns aspectos que se revelaram interessantes para uma abordagem mais detalhada. Em seguida, são referidos sumariamente aqueles que poderão vir a ser objeto de futura investigação:

- Obtenção de maiores indicadores quantitativos relativos aos benefícios da utilização dos métodos de concepção enxuta;
- Utilização da metodologia em projetos que não utilização métodos ágeis;
- Analise de utilização da metodologia nos projetos em fase inicial e/ou que não possuam um backlog estruturado.

Por fim, este trabalho se limita a apresentar uma sequencia de atividades praticas, baseadas em *design thinking*. O principal objetivo é ser um material de apoio, ou até mesmo um passo-a-passo, para aqueles times que necessitam revalidar e reavaliar seu *backlog* existente. O presente trabalho não tem a pretensão de ser um guia definitivo de como os times devem (ou não) reavaliar seu *backlog*, mas sim se apresentar como uma alternativa possível aos métodos tradicionais existentes. É importante salientar que os times também devem ficar livres para modificar qualquer dinâmica aqui apresentada ou até mesmo adicionar novas dinâmicas de acordo com as suas necessidades.

Referências

- [1] A. v. B. A. C. W. C. M. F. J. G. J. H. A. H. R. J. J. K. B. M. R. C. M. S. M. K. S. J. S.-l. Kent Beck, Mike Beedle and D. Thomas. “Manifesto for Agile Software Development”, 2001. [<http://www.agilemanifesto.org>; acessado 2017-10-30].
- [2] L. Bastos. “Da Descoberta do Ágil ao Manifesto”, 2003. [<http://pt.slideshare.net/lucabastos/manifesto-luca-bastos-agilevale-2013>; acessado 2017-10-30].
- [3] M. Fowler. “The New Methodology”, 2005. [<http://www.martinfowler.com/articles/newMethodology.html>; acessado 2017-10-30].
- [4] P. Caroli. *Direto ao ponto*. Casa Do Código, 2015.
- [5] M. Vianna. *Design thinking: inovação em negócios*. MJV Press, 2012.

- [6] Y. Monden. *Sistema Toyota de Produção: Uma Abordagem Integrada ao Just in Time*. Bookman Editora, 2015.
- [7] M. P. Tom Poppendieck. *Len Software Development - An Agile Toolkit*. Addison Wesley, 2003.
- [8] D. Council. “Eleven lessons: managing design in eleven global brands”, 2005. [[https://www.designcouncil.org.uk/sites/default/files/asset/document/ElevenLessons_Design_Council\(2\).pdf](https://www.designcouncil.org.uk/sites/default/files/asset/document/ElevenLessons_Design_Council(2).pdf); acessado 2017-10-30].
- [9] F. Castro. “What is OKR?”, 2015. [<http://felipecastro.com/en/okr/what-is-okr/>; acessado 2017-10-30].
- [10] F. Castro. “The Silicon Valley way of setting goals: an introduction to OKR”, 2015. [<https://medium.com/the-alignment-shop/introduction-to-okr-9912085830f0>; acessado 2017-10-30].
- [11] J. Doerr. *Measure What Matters: How Bono, the Gates Foundation, and Google Rock the World with Okrs*. Portfolio, 2017.
- [12] P. C. Lourenço P. Soares. “O método da cebola por Lourenço Soares”. [<http://www.caroli.org/o-metodo-da-cebola-por-lourenco-soares/>; acessado 2017-10-30].