Como lidar com prazos em projetos de software

Lucas Colucci
Agile Consultant

plataformatec

Table of Contents

Sobre o autor

Lucas Colucci

Bacharel em ciência da computação pela USP, mestre em gerenciamento de software pela Carnegie Mellon University, e Agile Consultant na Plataformatec. Foi voluntário em 3 Agile Conferences nos EUA, foi Scrum Master em uma empresa de mídia americana, publicou um artigo sobre métodos ágeis no WBMA. Nas horas vagas faz cheesecakes para os amigos.

Introdução

Durante os últimos anos, temos visto que problemas com prazos em projetos de software são recorrentes. Por esse motivo, tomamos a iniciativa de escrever esse ebook sobre como lidar com essa questão em diferentes estágios de um projeto e de diferentes formas.

Para facilitar a leitura, dividimos as informações aqui presentes da seguinte maneira:

1. Análise do problema: Nesta parte informaremos o que de fato é o problema de prazos em projetos de software, dando ferramentas para você decidir se você tem este problema ou não.
2. Pontos de Ação: Nesta parte apresentaremos como resolver o problema sob três diferentes perspectivas:
    - Melhora na definição do prazo;
    - Melhora na sua previsão de entrega;
    - Melhora no processo de desenvolvimento.
3. Monitoramento: Nesta parte daremos dicas de como monitorar o seu processo para garantir que as mudanças efetuadas estejam sendo efetivas dentro da sua empresa.

1. Análise do Problema

Imagine um projeto de software na seguinte situação:

Podemos definir o problema de prazo como a diferença entre a data de previsão de entrega e a data de deadline. Então no exemplo acima, o problema seria a diferença de 4 semanas entre o deadline do projeto e a previsão de entrega.

No entanto, será que isso é um problema mesmo? Digo, 4 semanas de atraso no projeto trarão conseqüências suficientemente relevantes para a empresa?

A seguir apresentamos alguns pontos nos quais você deveria se basear para responder às perguntas anteriores:

- Custo de atraso;
- Problemas emocionais;
- Problemas no produto.

1.1 Implicações do problema

1.1.1 Custo de Atraso

A primeira coisa para pensar quando nota-se que um projeto está atrasado é: quanto perderei com isso?

Para conseguir responder melhor a essa pergunta, apresentamos aqui o conceito de custo de atraso (Cost of Delay, CoD, em inglês).

CoD é o impacto econômico do atraso de um projeto. Em outras palavras, é o quanto você deixa de ganhar caso não tenha o produto ou a feature pronta até determinada data.

Um modo de chegar nesse número seria calculando o lucro semanal projetado para cada após o produto estar pronto.

Um jeito prático de analisar o CoD é estudar o padrão que ele segue em nosso projeto. Existem quatro diferentes curvas que prevêem o CoD de um projeto e você pode analisar em qual delas seu projeto se encaixa: padrão, urgente, data fixa e intangível.

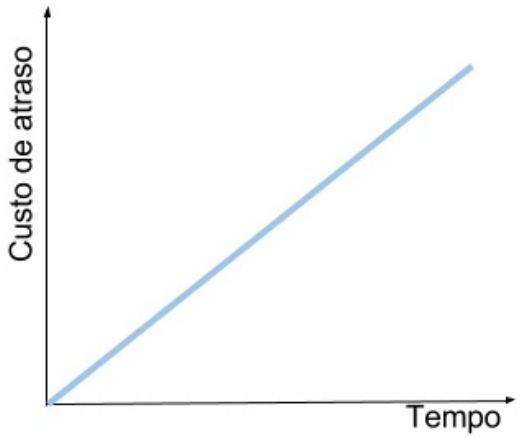
DICA

Custo de atraso pode ser também utilizado para priorizar projetos diferentes, diminuindo o custo de oportunidade total. Para isso realiza-se os seguintes passos:

1. Calcula-se o lucro semanal que cada implementação teria após finalizada;
2. Calcula-se aproximadamente quanto tempo demoraria para implementar cada uma;
3. Divide-se o lucro pela duração estimada do projeto.

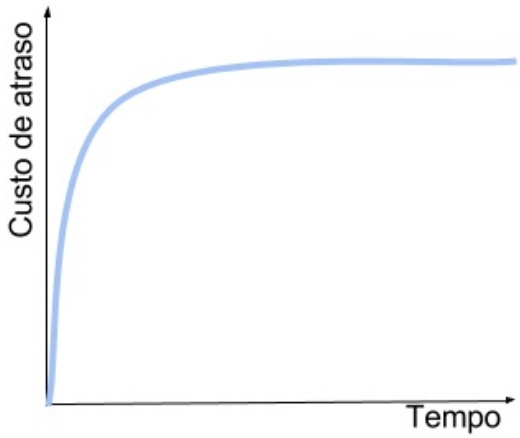
O coeficiente resultante é chamado de CD3 (cost of delay divided by duration). Quanto maior for o CD3 do projeto, mais importante ele é, pois dará retorno mais rápido.

Curva padrão



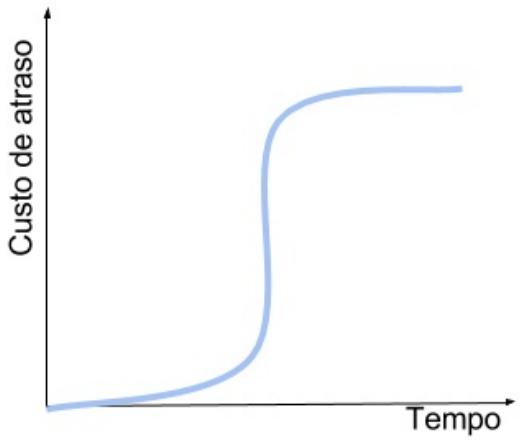
Na curva padrão, o CoD do projeto cresce de maneira praticamente linear com o tempo, portanto não existe uma data fixa na qual o projeto se torna muito custoso. Com isso, é possível calcular mais facilmente o custo de atraso semana a semana.

Curva Urgente



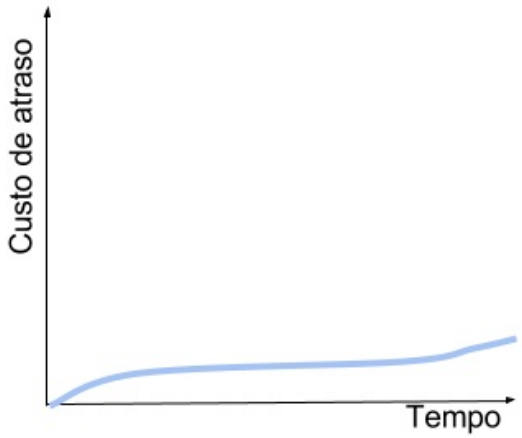
Na curva urgente, o projeto precisa acabar muito rápido para gerar algum valor, caso contrário um valor muito alto será perdido. Isso acontece, por exemplo, quando tem-se uma notícia de que um concorrente lançou um produto igual ao que está sendo desenvolvido e não há muito tempo para reagir e competir no mercado.

Curva de Data Fixa



A curva de data fixa é um exemplo clássico de projeto no qual o prazo é imutável. Neste tipo de projeto, o CoD está altamente ligado a uma data específica. O exemplo da curva urgente vale aqui também caso saiba que certo concorrente vai lançar seu produto em determinada data. Outro exemplo é quando existe uma campanha publicitária de âmbito nacional, ou até mesmo uma nova legislação, com data definida, e é imprescindível ter o produto pronto até lá.

Curva Intangível



A curva intangível cobre aqueles projetos no qual o CoD é muito pequeno. Projetos que seguem esse padrão geralmente tem baixa prioridade dentro das empresas pois causam pouco impacto nos resultados. Nestes projetos é possível ter um prazo de entrega muito mais flexível, já que valor que se perde com o atraso é baixo.

Apesar destas definições, um projeto pode ter comportamentos diferentes durante o seu desenvolvimento. Para ilustrar isso, digamos que um projeto que entrega um produto nunca antes visto pela sociedade esteja sendo desenvolvido e, devido a não existir concorrentes no mercado, segue a curva padrão. No entanto, após um ano de desenvolvimento, descobre-se que um concorrente está desenvolvendo um produto semelhante e que vai lançá-lo no dia X. A partir de agora, a curva segue o padrão data fixa.

1.1.2 Problemas Emocionais

Além da perda econômica de se atrasar um projeto, há também uma perda um pouco mais abstrata. Quando um projeto está atrasado, as pessoas envolvidas no mesmo tendem a se desmotivar, efeito conhecido como “Death March”, fazendo com que estas se perguntem se isso é culpa delas ou que questionem a liderança do projeto.

A desmotivação não exclui pessoas com cargos altos na companhia. O próprio CTO pode se desmotivar e se preocupar com a conversa que terá com o CEO para informar que o projeto atrasará.

É importante, portanto, saber qual é a cultura de sua empresa para prever estes tipos de reações. Isso pode influenciar em nossas decisões de prazos pois com uma equipe desmotivada após a falha de um projeto, a probabilidade de falhar novamente em outro aumenta pois o rendimento individual cai muito.

No entanto, caso a cultura da empresa seja aberta no quesito “falhas”, esclarecendo que elas podem servir como aprendizado, culpando o processo e não as pessoas quanto aos fatos ocorridos, esta questão pode ser minimizada ou descartada.

1.1.3 Problemas no Produto

Outro ponto importante a se pensar é a implicação que este atraso pode ter no produto final. Dependendo de como esse atraso é contornado, a qualidade do produto pode ser afetada para tentar melhorar a vazão do projeto. No entanto, essa queda de qualidade pode ser o que define o sucesso ou o fracasso de um produto. Além disso, uma falha em atender os requisitos do consumidor pode manchar a marca da empresa.

1.2 Analisando as implicações do problema

É importante lembrar que caso seu projeto esteja atrasado, é necessário pensar nas implicações que esse atraso pode ter no produto pois pode valer mais a pena entregar uma solução atrasada, mas que os consumidores precisam e gostam, do que um produto incompleto ou ruim no tempo certo.

Resumindo, é necessário olhar para o custo de atraso do projeto, para a cultura da empresa e para as possíveis divergências entre a ideia do produto e o entregado para avaliar se o atraso do projeto é ou não um problema grave.

2. Pontos de Ação

O problema de prazo pode ser definido como a diferença entre a data de previsão de entrega e a data de deadline. Pensando dessa maneira é possível dividir a solução de duas maneiras: ou muda-se a data prevista ou o deadline.

A data prevista pode ser modificada melhorando os processos ou mudando o método de previsão. Com isso em mente, dividimos esta seção em:

- Melhoria na Previsão;
- Definição de Prazo;
- Melhoria no processo.

2.1 Melhoria na Previsão

Antes de concluir que seu projeto está atrasado, primeiro pode ser necessário melhorar como você está projetando o término do trabalho. A seguir mostramos alguns dos métodos de previsão que a Plataformatec tem utilizado e explicamos por que mudamos de um para o outro até chegarmos no método de Simulações Monte Carlo.

Estimativas sem dados

Alguns times preveem o fim de um projeto a partir de estimativas baseadas no conhecimento das pessoas envolvidas e não em dados reais de projetos. Um problema desse método é que ele é altamente enviesado. Uma pessoa pode entender melhor um

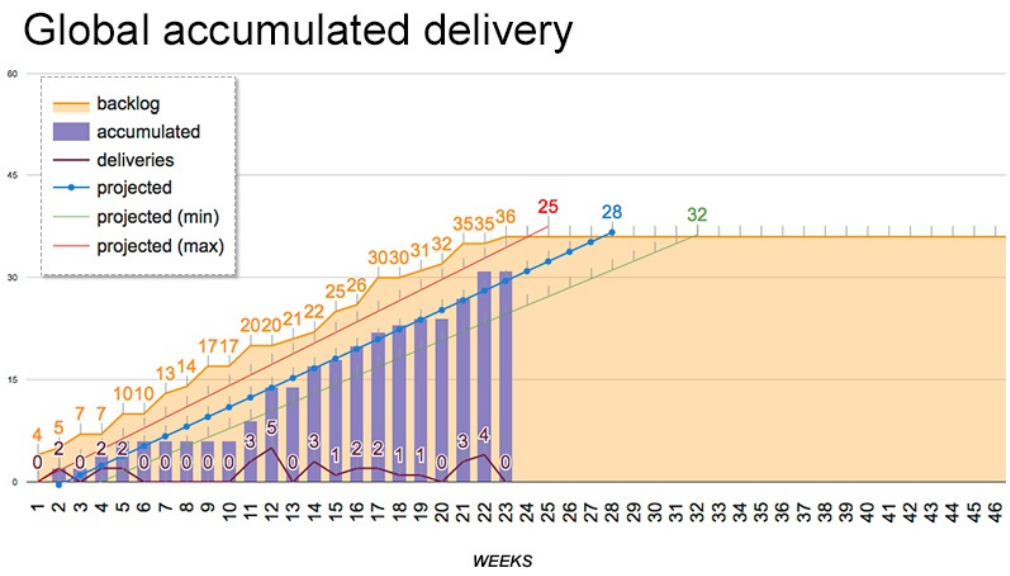
problema do que outra, ou pode ter uma argumentação melhor na hora de convencer o resto do grupo de que sua estimativa é melhor, ou pode até maquiar sua estimativa com medo de retaliação caso a data projetada esteja muito longe do deadline.

Vazão Média

O primeiro método que utilizamos foi a predição por vazão média. Este é um pouco mais preciso e é baseado em algum dado concreto. No entanto, como explicamos em nosso blog post "Power of the metrics: Don't use average to forecast deadlines", projetos raramente possuem uma distribuição de vazão que façam a média, mediana e moda caírem próximas do mesmo valor. A distribuição desses valores não segue uma distribuição Normal e sim de Weibull, o que faz com que a vazão média seja uma métrica arriscada para projeções de entrega.

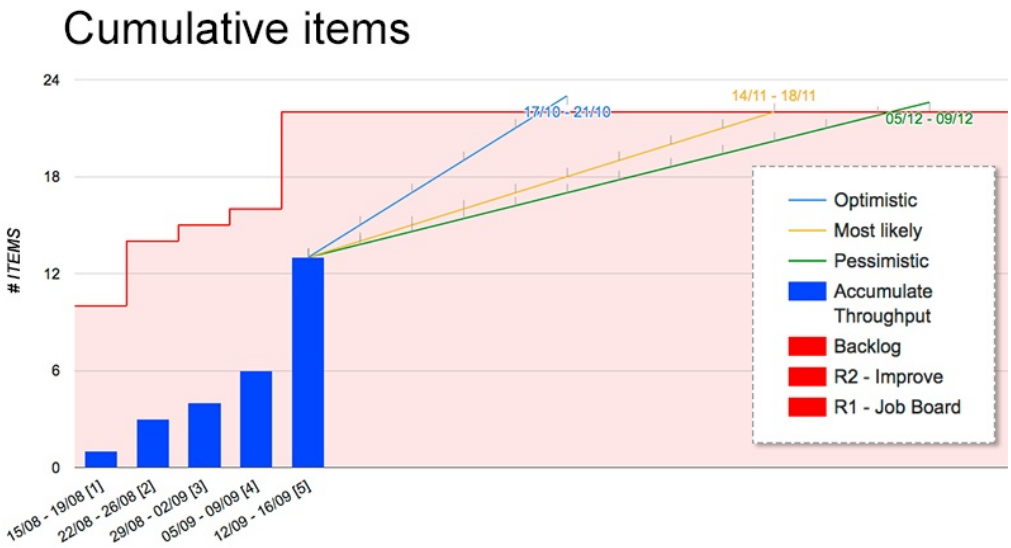
Regressão Linear

Após vermos os pontos fracos da vazão média, passamos a usar regressão linear para obter mais precisão em nossas predições. E de fato, a precisão aumenta um pouco por levar em consideração o seu histórico passado e por pesar um pouco melhor as ocorrências passadas. No entanto, como falamos em nosso blog post "Looking at Lead Time in a different way", o método de regressão possui alguns pré-requisitos, e um deles é que os dados sigam uma regra chamada em inglês de "homoscedasticity", que significa que, em relação à reta de regressão linear, os valores precisam seguir uma variação constante. E isso costuma não acontecer com as vazões de um projeto de software.



Análise de Cenário

Com a intenção de corrigir o erro estatístico de se usar regressão, passamos a utilizar a análise de cenário. Esse método foi muito utilizado dentro da Plataformatec pois é um método manual no qual analisamos os percentis do histórico de vazão e projetamos retas a partir desta análise.

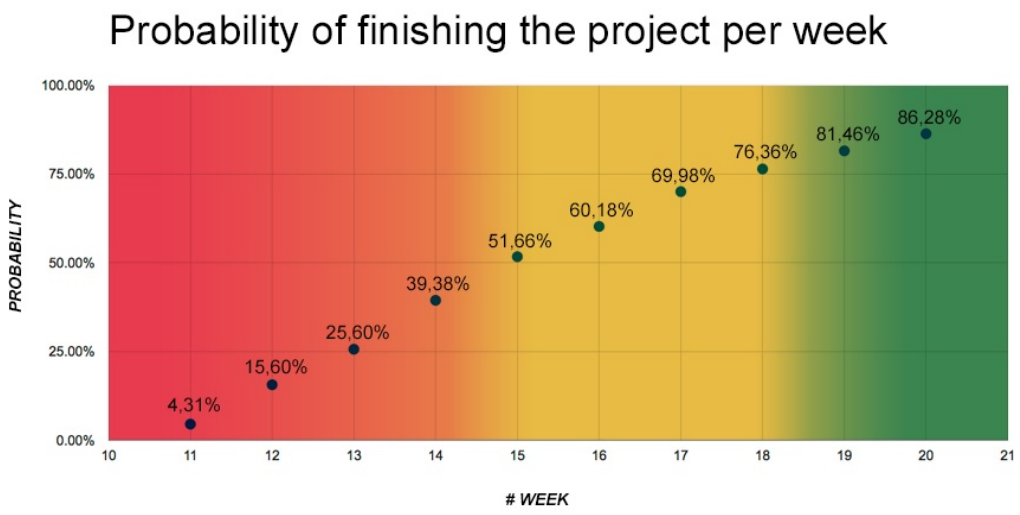


No entanto, assim como nos métodos passados, o modelo não tratava o fato de que o backlog de tarefas cresce juntamente com a vazão acumulada, como podemos ver no gráfico da regressão linear (retirado de um projeto real).

Para corrigir esse viés, começamos a utilizar o método de Monte Carlo para termos mais informação sobre a data de término do projeto.

Monte Carlo

O mais interessante sobre este método é que ele não da apenas uma semana em específico como previsão de entrega. O resultado é a probabilidade de se terminar o projeto semana a semana, o que nos diz muito mais do que uma única data de projeção. Para entender um pouco mais sobre Simulações de Monte Carlo, sugerimos a leitura de nosso blog post “Forecasting software project’s completion date through Monte Carlo Simulation” no qual explicamos como o método funciona e disponibilizamos gratuitamente uma planilha para você começar a utilizar em seu projeto.

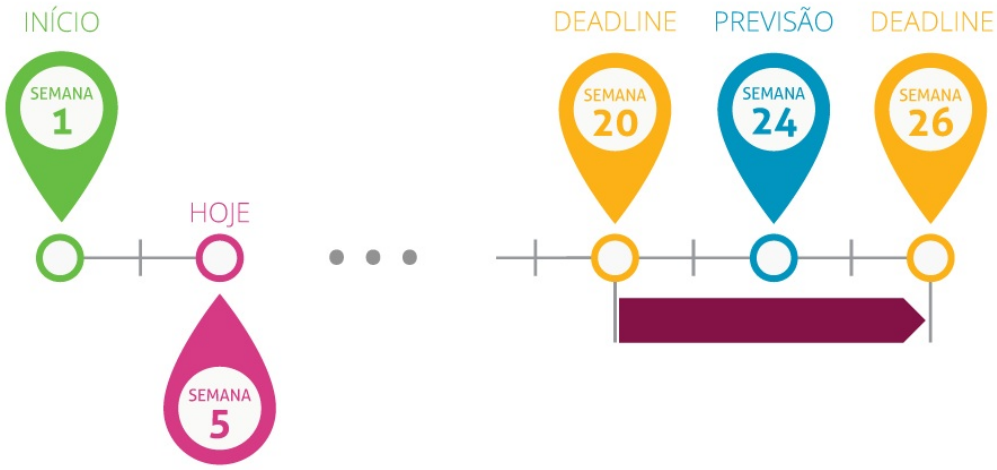


O que se tira de conclusão do gráfico de probabilidade são áreas onde temos maior ou menor probabilidade de entregar o projeto. Quando testamos o método em projetos passados, decidimos que a área com valores maiores do que 75% expressa uma alta probabilidade de entrega. Entre 50% e 75% consideramos que a probabilidade de entregar é baixa, no entanto possível, se conseguirmos melhorar o nosso processo. E menos do que 50% consideramos muito difícil e medidas mais drásticas precisam ser tomadas caso haja a necessidade de entregar nessas semanas.

Se você não está utilizando simulações de Monte Carlo ainda, sugerimos que comece, pois trará muito mais assertividade nas suas predições. Com essa nova ferramenta em mãos, você poderá verificar se seu projeto está atrasado ou não e definir ações de acordo com a probabilidade de entrega na semana do deadline.

2.2 Definição de Prazo

Esta ação é mais indicada antes de começar um projeto ou nas primeiras semanas de desenvolvimento, pois é quando normalmente há mais flexibilidade para mudanças de prazos.



Na nossa experiência, já vimos diversos motivos para uma definição de prazo imprópria em um projeto, no entanto a mais recorrente é a falta ou falha de comunicação entre o CTO e o CEO. Com isso, vamos focar em como melhorar essa comunicação para que seja possível alinhar as decisões de negócio com as decisões de tecnologia.

Sabe-se que a área de desenvolvimento de software segue processos que diferem não só do resto da indústria mas muitas vezes difere também das outras áreas na mesma empresa. A quantidade de incertezas são, geralmente, maiores e faz com que o tempo para terminar um projeto seja altamente variável mesmo quando, aparentemente, tenha natureza conhecida. E essa diferença precisa ser entendida por ambos CTO e CEO.

EXEMPLO

Um exemplo que pode ilustrar muito bem o problema de alinhamento entre C-Levels é no caso em que um CTO precisa lidar com dois tipos de softwares diferentes em sua empresa: um produto novo e um software legado.

Quando desenvolve-se um produto novo, suas estimativas iniciais são pouco precisas, mas devido à implementação de ciclos de feedback rápido, é possível mudar o caminho ao longo do desenvolvimento fazendo com que, mesmo que com estimativas erradas, os resultados do desenvolvimento apareçam rapidamente. No entanto, quando o projeto envolve software legado, o processo

> é um pouco mais demorado devido à carga de conhecimento necessária para efetuar novas mudanças no sistema, e também devido ao rigor necessário em testes de qualidade para mudar algo que já esteja em produção.
>
> É crucial que o CEO tenha esse conhecimento em mente quando define suas estratégias de negócio, caso contrário ele pode considerar que um projeto de evolução de sistema legado seja semelhante ao desenvolvimento de um sistema novo, o que causaria uma definição de prazo errônea.

As causas desse desalinhamento podem ser:

1. **Falta de comunicação** quando decisões de negócio e tecnologia são tomadas. Essas duas decisões precisam estar em constante harmonia pois uma é o que impulsiona a outra.
2. **Falta de priorização** das demandas. Às vezes existem muitas features para serem desenvolvidas de uma vez, sem qualquer priorização. Isso faz com que o processo de desenvolvimento fique saturado e demore para dar vazão a toda essa demanda.
3. **Conflito de mentalidade**. Devido aos diferentes backgrounds dos envolvidos e, portanto, diferentes visões sobre a empresa, a relação entre CTO e CEO pode se tornar deficiente e conflitante. Isso impede a evolução do processo por não haver acordos.

Por mais que seja possível dar um manual com um passo-a-passo do que se deve fazer para melhorar essa relação, tudo se resumiria a uma ação: melhorar a comunicação.

Quando você se encontra em posições de alta responsabilidade, você precisa ter um entendimento maior dos seus pares e o trabalho que estes exercem para que consigam, juntos, traçar o futuro da empresa. Por isso, para melhorar a comunicação, é necessário entender melhor como funciona a tomada de decisões do CEO, quais são as estratégias da empresa para os próximos meses e anos e porque elas são importantes.

Com isso feito, é necessária a explicação de como um processo de desenvolvimento de software difere de outros processos, e como lidar com isso da melhor maneira possível. Com ambas as partes a par dos caminhos que precisam ser tomados, é possível alinhar prazos de modo mais fácil.

2.3 Melhoria no Processo

Caso você já tenha tentado alinhar melhor o prazo e suas predições já estejam suficientemente precisas, entra a parte de melhorar seu processo. A ideia é que aos poucos sua vazão aumente e sua equipe consiga entregar todo o trabalho até a data estipulada.

2.3.1 Visibilidade

O primeiro passo para melhorar seu processo é aumentar a visibilidade do mesmo. Isso é inclusive uma prática do Kanban. Sem visibilidade, não é possível medir se a vazão está realmente aumentando e se as ações que estão sendo tomadas estão tendo efeito. Além disso, uma visibilidade maior ajuda a encontrar problemas mais rapidamente, como gargalos no processo.

COMO FAZEMOS?

Para a Plataformatec, ter visibilidade nos ajuda a tomar decisões e gerar ações para melhorar nosso processo.

Para isso, fazemos o seguinte:

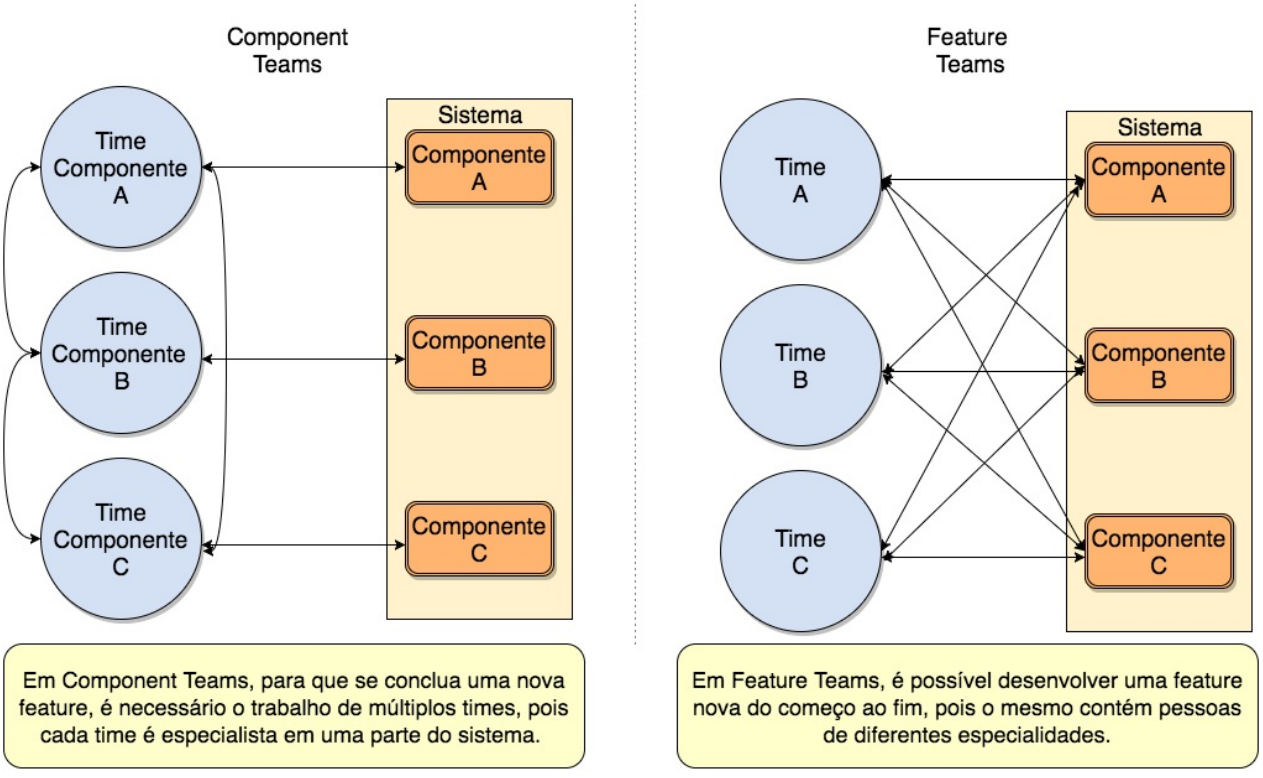
- **Monitoramos** nossas métricas para acompanhar a saúde de nosso processo.
- **Analisamos** se as métricas estão fora do esperado e, se estiverem, precisamos tomar alguma medida.
- **Agimos** mandando relatórios semanais com o resumo das métricas e nossas sugestões para melhoras no processo.

Algumas das ações que podem ser tomadas a fim de aumentar a visibilidade no projeto são:

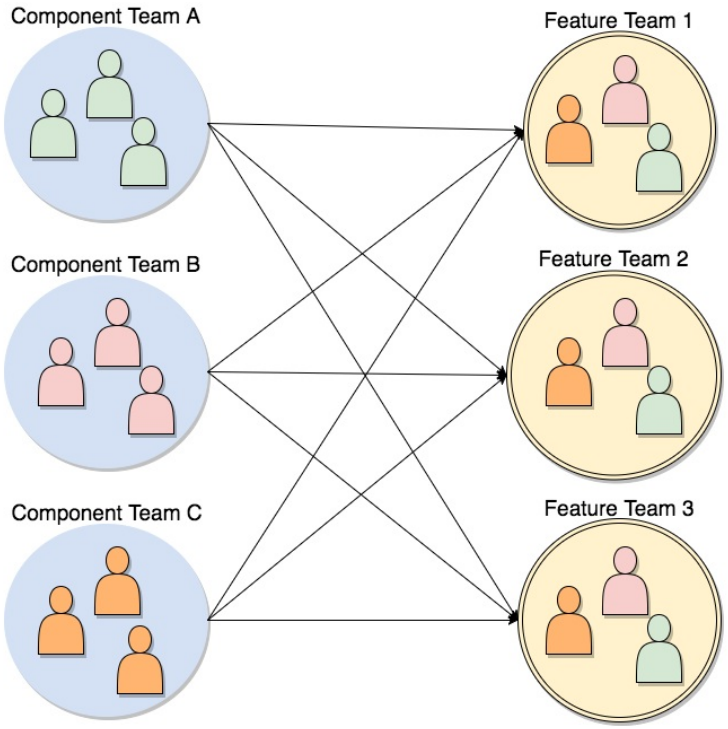
- Utilização de um quadro Kanban
- Limitar Work in Progress do sistema (temos um blog post a respeito disso: Case Study of a WIP Limit Implementation: Why, When and How to use WIP Limits)
- Reunir algumas métricas do sistema para gerar gráficos como:
  - Lead time breakdown (mais informação em nosso blog post "Why we love metrics? Learning with Lead time")
  - Cumulative Flow Diagram (mais informação em nosso blog post “Why we love metrics? Cumulative Flow Diagrams”)
  - Burnup charts (mais informação em nosso blog post "Why we love metrics? Throughput and Burnup charts")

2.3.2 Component Teams x Feature Teams

Com seu processo visível, agora você pode solucionar alguns problemas que a maioria dos times possuem. Um dos problemas que vemos muito no mercado é o reflexo da estrutura dos times na arquitetura dos seus sistemas (este fenômeno é conhecido como Lei de Conway). Na intenção de minimizar a interferência externa, porções do software são isoladas em componentes, muitas vezes desnecessariamente.



Em Component Teams, o problema é que uma feature que precise ser desenvolvida vai passar por inúmeros times diferentes, que geralmente não se comunicam rapidamente, atrasando o desenvolvimento do produto final. O desenvolvimento de uma feature pode envolver times de diferentes componentes de um sistema. Por exemplo, em um sistema de vendas online é possível ter um time para desenvolver a parte de pagamentos, um time cuidando do sistema de inclusão de itens, um time cuidando das páginas web, e etc.



O que tem surgido como solução para isso é o que muitos chamam de “feature teams”. Estes têm como tarefa fazer uma feature do começo ao fim, envolvendo todos os diferentes sistemas necessários. Desta maneira entrega-se valor mais rápido e evitando custos de comunicação entre times para o desenvolvimento de uma única feature.

Uma opção para construir feature teams é, inicialmente, dissolver cada um dos times de componente em diferentes equipes para formar times com um leque de conhecimento mais abrangente.

2.3.3 Integrantes Altamente Especialistas

Com feature teams no lugar de component teams, existem diferentes tipos de especialistas dentro de um time só. No entanto para se obter 100% de aproveitamento da metodologia ágil, os integrantes da equipe precisam estar aptos a cobrir diferentes partes de um processo. Com isso, os gargalos de um sistema podem ser atacados por pessoas de diferentes especialidades caso necessário. Um desenvolvedor poderia pegar tarefas tanto de back-end quanto de front-end, por exemplo.

Para transformar um time de especialista para especialista generalista, pode-se tomar algumas ações:

- Rotação entre times;
- Rotacionar papéis dentro do time;
- Filas únicas ao invés de filas individuais;
- Incentivar code review.

Rotação entre times

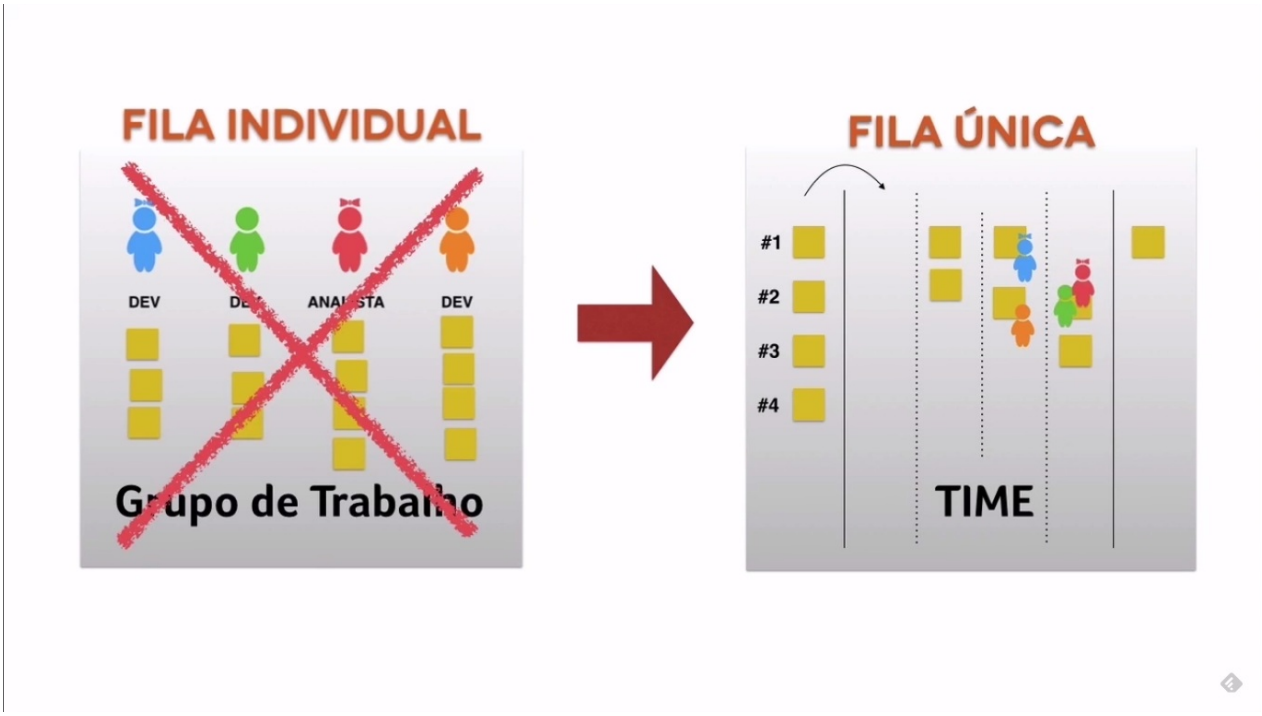
Rotação de pessoas entre times é muito útil para evitar silos de conhecimento. A ideia é tão simples quanto parece: de tempos em tempos, troca-se pessoas de projetos. Sua implementação pode ser um pouco mais complexa do que apenas tirar pessoas de seus projetos. Pode ser necessária alguma ferramenta de controle dessas ações para garantir que todos sejam rotacionados em um período semelhante de tempo e para times diferentes, não mudando apenas entre dois times.

Rotação de papéis

Caso você siga um processo estilo Kanban, no qual as decisões são tomadas on-demand, a troca de papel é praticamente natural. Sempre que move-se uma história, analisa-se o quadro kanban da direita para a esquerda e, com base nisso, atua-se no card na coluna necessária para entregar mais valor para o cliente, o que pode implicar em trocar de papel quando possível.

Filas Únicas

Um dos erros mais comuns em times de desenvolvimento é cada membro possuir sua própria fila de tarefas. Isso acaba fazendo que tarefas que aquela pessoa faz com mais facilidade sejam direcionadas a ela. No entanto esse tipo de estrutura aumenta muito a variabilidade do lead time de uma tarefa. Uma melhor maneira seria usar uma única fila para todo o time, assim a variação no tempo de completude de uma tarefa diminuiria e todos os indivíduos puxariam tarefas diferentes, sem se importar se é especialista em tal tarefa ou não.



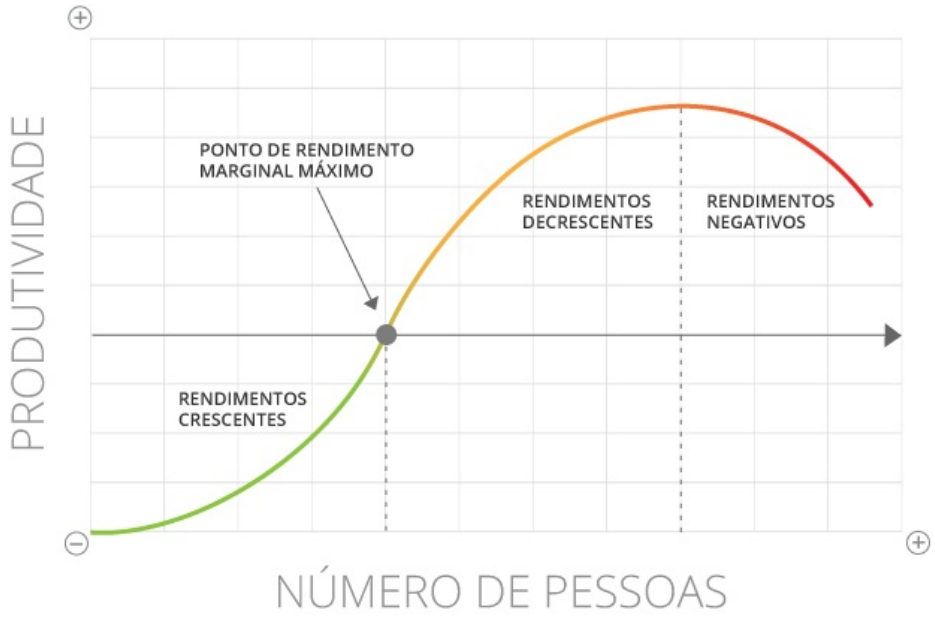
Code Review - Pull Requests

Uma maneira mais técnica, mas que também ajuda a disseminar o conhecimento do time, é a prática da revisão de código por meio de pull requests. Um desenvolvedor, quando quer adicionar alguma nova funcionalidade ao sistema, primeiramente apresenta suas mudanças em um ambiente separado e o resto do time avalia essa nova mudança, tecnicamente e funcionalmente. Com isso todos os integrantes do time tem contato com diversas partes do sistema e o resultado é um código revisado antes de entrar em produção.

Além disso, essa técnica é um modo de operacionalizar uma prática importante do XP (Extreme Programming), o código coletivo.

Falta de Mão-de-Obra

É importante ressaltar que um processo de produção, seja de software ou qualquer outro produto, geralmente segue uma curva que segue a Lei dos Retornos Marginais Decrescentes. A intenção aqui não é entrar em detalhes de Microeconomia, mas mostrar a ideia para que você entenda o que precisa ser feito.



O aumento de produtividade de um processo não se mantém diretamente proporcional de acordo com o aumento no número de pessoas. Isso porque há um número limitado de modos de obter maior especialização do trabalho e porque cada trabalhador adicional provoca efeitos de superpopulação (McGuian, James et. al., Economia de Empresas, p.167-172).

É importante lembrar que todo time possui um limite máximo de pessoas, e é crucial saber se você já atingiu esse limite. Quando uma pessoa nova é adicionada a um time, os seguintes cenários podem ocorrer:

1. O time pode realmente estar precisando desse novo integrante e, portanto, a vazão cresce tanto que o retorno do investimento é rapidamente alcançado. (Rendimentos Crescentes)
2. O time ainda consegue absorver mais uma pessoa, no entanto o ganho na vazão dessa inclusão é menor que no cenário 1. (Rendimentos Decrescentes)
3. O time está saturado e o novo integrante faz com que as pessoas tenham que disputar tarefas e investir tempo excessivo na coordenação do trabalho, o que causa uma queda na vazão do time. (Rendimentos Negativos)

No entanto, a curva da lei dos retornos marginais decrescentes tem como premissa que você fixou as outras variáveis de entorno do seu sistema, e que está mexendo apenas no número de membros. Portanto, para continuar desfrutando de retornos marginais positivos, é necessário que você modifique outras variáveis do seu sistema além da adição de mais pessoas.

EXEMPLO

Para que seja possível visualizar mais facilmente o que significa a saturação de pessoas em um time, imagine o seguinte exemplo. Carlos trabalha em um carrinho de cachorro-quente em uma rua movimentada.

Ele cuida tanto da montagem do cachorro-quente quanto da venda do mesmo. Por ter uma fila muito grande de pessoas esperando pelo seu hot-dog, ele vê que algumas vezes as pessoas saem da fila. Com isso em mente ele decidiu contratar mais uma pessoa para cuidar da parte de montagem enquanto ele cuida das vendas.

Após a entrada do novo integrante, a venda de cachorro quentes aumentou em 100% pois ninguém mais saia da fila (Exemplo de rendimento crescente).

Carlos, no entanto, acredita que mais um integrante ainda traria benefícios ao negócio, pois a nova pessoa poderia recolher o resto de comida que fica em torno do carrinho e organizar os ingredientes para que o montador ficasse focado apenas na montagem. Assim ele poderia eliminar a fila. Carlos então contratou mais uma pessoa e sua venda de hot-dogs aumenta em 25%, ao invés dos 100% conseguidos com o primeiro novo integrante (Exemplo de rendimento decrescente).

Carlos ainda vê que o montador de hot-dogs gasta muito tempo customizando o final da montagem, pois algumas pessoas gostam de mostarda e ketchup e outras não. Pensou então em trazer ainda mais uma pessoa para o time, para

> cuidar apenas da customização dos produtos. No entanto, quando a pessoa chegou ele viu sua produtividade diminuir em 30%, e notou que o que acontecia era que o novo integrante tinha que dividir espaço com o montador e, portanto, atrapalhava mais do que ajudava no fluxo (Exemplo de rendimento negativo).
>
> No entanto, Carlos era um homem de visão, queria crescer seu negócio. Para acomodar as 4 pessoas e aumentar ainda mais o fluxo, ele decidiu abrir uma loja de cachorro-quente. Nessa loja agora, ele tem espaço para todos e ainda conseguiu colocar mais duas pessoas como atendentes. (Exemplo de mudanças no sistema para manter um rendimento crescente).

Em um projeto de software, as modificações necessárias incluem todas as melhorias que já falamos neste ebook. Mas além delas, existem alguns fatores extras que precisam ser garantidos para que a adição de uma nova pessoa no time seja efetiva:

- **Alta qualidade de código:** caso a qualidade do código esteja ruim, a adição de um novo membro em nada mudará pois a curva de aprendizado para desenvolver novas funcionalidades será grande e o seu lead time só aumentará com o tempo devido à complexidade gerada pela baixa qualidade.
- **Arquitetura de software:** a arquitetura do software precisa possibilitar que a adição de um novo integrante melhore o throughput global.
- **Organização dos times:** os times precisam estar corretamente organizados para que a adição de um novo membro não desestabilize o funcionamento do fluxo.

Com isso, é necessário que antes de tomar a decisão de contratação ou deslocamento de alguém, você se atente às condições nas quais seu time se encontra para saber se você terá rendimentos positivos através da adição de um novo membro ou não.

2.3.5 Priorização

A partir da priorização, é possível garantir que o que está sendo entregue, iteração após iteração, é o que mais traz valor para a empresa naquele momento. Em casos onde nenhuma ação deu certo e seu produto final atrasou, ainda existe a certeza de que o que foi feito até ali foi o mais importante que tinha a ser feito, e é possível cortar eventuais features que não entreguem tanto valor.

No entanto, quando falamos de priorização, muitos têm dificuldade em encontrar critérios para tal. Por isso, resolvemos elencar o conjunto de características de uma feature que definem seu valor:

- **Redução de risco:** caso a tarefa sendo desenvolvida venha a trazer alguma redução

de risco para o sistema, como aumento na cobertura de testes, refatoração do código para diminuir o risco de falhas e etc.

- **ROI:** caso a tarefa traga benefícios financeiros diretos para a empresa, é possível calcular sua prioridade baseado no retorno monetário que essa história vai dar em relação às outras.
- **CD3:** Como explicamos anteriormente, em alguns casos é válido calcular o cost of delay das features para analisar a importância de cada uma.

3. Monitoramento

Após aplicar todos os métodos possíveis para aumentar sua vazão, melhorar sua predição e definir melhor o prazo do projeto, é importante que você consiga monitorar seu andamento para que, caso alguma outra decisão tenha que ser tomada, que seja o quanto antes.

Além de manter as métricas e gráficos do projeto atualizados, é necessária a análise e divulgação de tempos em tempos de um resumo das métricas para que todos os times saibam como o projeto está andando e consigam atuar em possíveis gargalos no processo.

Outra boa prática que estamos adotando na Plataformatec é a realização de uma cerimônia chamada “Reality Check”. Esta reunião acontece quando temos a sensação de que o projeto pode estar saindo um pouco do controle e paramos tudo para entendermos onde estamos.

A prática é bem simples: definimos o que achamos que conseguimos entregar em cada semana que nos resta. Assim é possível alinhar as expectativas e monitorar como o projeto está andando. Esse tipo de reunião também deixa o time todo sincronizado, para que o processo volte a fluir como inicialmente.