Talita Pagani Follow
Mestre em Ciência da Computação | Consultora/Tester de Acessibilidade na Utilizza | Designer na Nexaas
Jan 8 · 7 min read

Como estimar o esforço de desenvolvimento de software e por que isso é tão difícil? — Parte 2

Em "*Como estimar o esforço de desenvolvimento de software e por que isso é tão difícil? — Parte 1*", apresentei uma pesquisa que realizei para entender os principais problemas que ocorrem no mundo do desenvolvimento com relação à estimativas.

Agora é hora de entrar na parte que me motivou a escrever esta série de posts: **compartilhar a minha experiência sobre esse assunto junto com algumas dicas**, algo que muitas pessoas já me pediram informalmente. Longe de ser uma técnica formal ou receita de bolo, são dicas sobre organização pessoal que podem ser aplicadas independente da abordagem que você já utiliza.

Elas são baseadas na minha vivência em diversos projetos da área de desenvolvimento ao longo desses 13 anos de atuação na área. A intenção não é esgotar este assunto que é tão amplo, tampouco apresentar algo que vai funcionar magicamente para qualquer situação em que é preciso fornecer estimativas.

Eu sempre fui muito preocupada em fornecer estimativas minimamente realistas e o conteúdo sobre gestão de tempo na especialização em gerenciamento de projetos me ajudou muito a refinar a perspectiva sobre como estimar requisitos. Mas foi com a prática e com técnicas próprias que consegui me organizar para não errar feio e "queimar meu filme" quando preciso fornecer estimativas (não que às vezes eu não dê umas derrapadas).

Tem uma frase bem batida que diz que "*não se pode controlar aquilo que não se pode medir*" atribuída ao Peter Drucker e quem tem várias adaptações. Com relação a estimativas, isso faz sentido em vários momentos, visto que é frequente desenvolvermos recursos ou

funcionalidades que são semelhantes a algo que já foi desenvolvido anteriormente. Ter essas métricas ajudam também a acompanhar a nossa evolução enquanto profissionais.

Documente suas tarefas para criar uma base de conhecimento

Como você irá lembrar quanto tempo leva para fazer um determinado recurso se você não tem isso documentado? Possivelmente, são nesses momentos que entram os chutes nas estimativas, o famoso "*olha, se não me engano, leva X tempo pra fazer isso, mas eu tenho que dar uma olhada aí*".

Pedro (Antônio Fagundes) da série "Carga Pesada" falando a famosa frase "É uma cilada, Bino!"

Uma estratégia que adoto desde 2012 para me ajudar nas estimativas é documentar a duração real de determinadas tarefas em uma planilha após tê-las realizado. Essa documentação não tem somente a duração, mas detalhes da tarefa, o que precisava ser entregue, qual era o requisito e quais problemas ocorreram.

Ter uma base histórica do que já fizemos, quanto esforço dispendemos e como fizemos ajuda também a não reinventar a roda, até mesmo porque não dá pra confiar só na memória.

Na imagem a seguir, apresento o modelo que usava para documentar as atividades de QA, quando atuei como analista de testes, uma função nova para mim na época na qual eu não tinha experiência.

Documentar as tarefas que eu fazia começou a me ajudar muito quando precisava fornecer estimativas, mesmo para atividades novas. A estrutura da planilha pode variar muito de acordo com sua função e as suas necessidades. Para front-end, eu coloco outros atributos e para análise de negócios e requisitos (quando atuei nessa função) eram outras informações a serem registradas.

Projeto	Tipo de Teste	Descrição do Teste	Condições de teste	Duração
Interno	-	Elaboração do relatório de testes	-	02:30
[redacted]	Integrado / Funcional	Teste de funcionamento das telas de submit	1. Verificar se está gravando a pontuação corretamente 2. Verificar se os dados não estão passando de uma atividade para outra 3. Verificar se está bloqueando a atividade depois de 3 tentativas 4. Verificar se as informações estão aparecendo na tela de "Detalhes"	03:00
[redacted]	Integrado / Funcional	Teste de funcionamento das telas de atividades	1. Verificar se o hover dos botões está correto e se o cursor está adequado 2. Verificar se os cliques e os eventos dos cliques estão corretos 3. Verificar se drag and drop estão corretos 4. Verificar se os selects estão funcionando 5. Verificar os scrolls	07:00
[redacted]	Integrado / Funcional	Teste de funcionamento completo com report simplificado	Mesmas condições dos casos 2 e 3	05:00
[redacted]	Funcional	Verificação dos valores das operações aritméticas de cada atividade do curso	1. Verificar se os valores / resultados apresentados nas dicas estão corretos 2. Verificar se o resultado da operação aritmética apresentada está correto 3. Verificar se não há operações repetidas em uma mesma tela 4. Verificar disposição dos números, do sinal da operação e dos resultados na tela 5. Verificar se ao digitar a resposta e clicar em "Conferir" está conferindo a resposta adequada	15:00
[redacted]	Funcional	Verificação dos valores das operações aritméticas de uma única atividade do curso	Mesmas condições do caso 5	00:40
[redacted]	Funcional / Usabilidade	Verificar se todas as funcionalidades apresentadas no escopo do teste contém alguma inconsistência, falha ou anomalia em seu comportamento	1. Verificar manipulação de livros 2. Verificar criação a alteração de páginas 3. Verificar organização e ordenação de páginas 4. Verificar exclusão de páginas 5. Verificar uso da ferramenta de texto 6. Verificar navegação entre páginas 7. Verificar manipulação de objetos 8. Verificar visualização de livros	21:00

Planilha de documentação das minhas métricas de teste com as colunas: projeto, tipo de teste, descrição do teste, condições do teste e duração da tarefa quando realizei, em horas. Há outras colunas que não estão sendo exibidas na imagem, como observações e data da primeira realização da atividade.

Essa estrutura é semelhante a de uma biblioteca de padrões (*pattern library*), onde documenta-se soluções de sucesso para problemas recorrentes em um determinado contexto, com a diferença de que aqui não há a preocupação de documentar um problema recorrente e suas soluções, mas sim fazer como se fosse um log de desenvolvimento.

Uma planilha de desenvolvimento front-end poderia ter a seguinte estrutura:

- Projeto/Produto
- Requisito ou Caso de Uso
- Descrição da implementação
- Framework utilizado
- Bibliotecas utilizadas
- Problemas encontrados
- Duração prevista
- Duração real (em horas)
- Prazo previsto
- Período real (em dias)

Se você realizar uma atividade semelhante em outro projeto/produto, vale a pena registrar novamente na tabela para ter um comparativo entre os tempos/prazos, se foram encontrados problemas diferentes, se você as durações reais se aproximaram das estimativas e se você agora consegue produzir a mesma funcionalidade de forma mais rápida com o passar do tempo.

> *Mas como eu contabilizo esse tempo? Vou ter que ficar com um timer, marcando no relógio?*

Não! Tem muitas ferramentas que podem te ajudar a automatizar essa contagem:

- **WakaTime:** contabiliza quanto tempo você programou por dia fornecendo métricas por linguagem, editor, sistema operacional e, se você usa GitHub, por projeto do GitHub. Integra com diversos editores e até com o navegador;

- **RescueTime:** para quem não trabalha com código (e pra quem trabalha também), o RescueTime mede quanto tempo quando você passa em aplicações e sites, categorizando cada aplicação e classificando entre ferramentas produtivas, neutras ou improdutivas. É um ótimo recurso para acompanhar sua produtividade também;

- **Traxmo:** É uma aplicação web de gerenciamento de projetos gratuita para usuários únicos que permite criar projetos e tarefas da forma como quiser. Para as tarefas, você pode inserir o tempo manualmente ou usar um cronômetro, o que é muito útil. Essa ferramenta eu uso constantemente quando faço freelas.

Essas ferramentas também ajudam a medir quanto tempo, em média, você passa codificando ou usando as ferramentas técnicas do dia-a-dia, permitindo que você saiba o seu tempo produtivo diário e use isso como parâmetro para estimativas futuras. Se você programa cerca de 3,5 horas por dia, uma tarefa que leva 8 horas pode ser estimada com um prazo de 3 dias.

Tem alguma outra sugestão? Deixe nos comentários :)

Organize a forma como você faz os commits para ajudar em estimativas futuras

Você pode resgatar o histórico de commits de uma funcionalidade para estimar ao menos o prazo de realização de uma funcionalidade semelhante a outra que você já fez. Mas isso vai depender uma boa estruturação dos commits e das mensagens de commit.

Eu costumo aplicar a técnica de commits atômicos. Eles facilitam a revisão de código, a reversão caso algo tenha sido mal-sucedido ao longo do desenvolvimento da funcionalidade e facilitam no momento de descrever os commits. Traduzindo o exemplo desse post de Nathan

Rambeck, uma série de commits atômicos (e suas respectivas mensagens) no desenvolvimento de um formulário poderia ser:

- Criado o HTML para o formulário
- Estilização do formulário
- Adição de validação HTML5
- Correção de alguns erros de JS
- Adição da submissão do formulário via AJAX com resultados do servidor 'mockados' do PHP
- Adição de validação JS ao formulário
- Log dos resultados do formulário no banco de dados
- Estilização da validação de formulário e de mensagens de erro/sucesso

Se alguém perguntar como fazer um formulário semelhante, você tem registros disso. Se alguém pedir um prazo para estilizar um formulário e as mensagens de validação, você tem registros disso também. Triangulando as informações desses commits com os registros do WakaTime, por exemplo, você consegue ter uma base para estimar tanto tempo quanto prazo através de uma estimativa por analogia.

Reduza riscos de potenciais atrasos com a estimativa de três pontos

Dar um prazo acurado não significa dar um prazo justo. Sempre podem acontecer imprevistos que acabam atrasando alguma tarefa. Por isso, a gente tem que contar com a famosa "gordurinha" nas estimativas que fornecemos já contando com alguns riscos e imprevistos. O problema é que colocar essa "gordurinha" pode ser subjetivo também.

Particularmente, gosto muito de usar a técnica PERT, também chamada de estimativa de três pontos. Ela consiste em calcular a duração média de uma tarefa utilizando uma estimativa otimista (O), mais provável (MP) e pessimista (P) aplicando o seguinte cálculo:

$$PERT = (O + 4 \times MP + P) / 6$$

Se você tem uma tarefa **A** com estimativa otimista de **12h**, mais provável de **17h** e pessimista de **28h**, o resultado seria uma média de **18h** para realizá-la:

$$A = (O + 4 \times MP + P) / 6 = (12 + 4 \times 17 + 28) / 6 = 18\ h$$

Bom, apenas uma hora de respiro pode não ser suficiente. Então você pode, se desejar, aliar ao cálculo também a variância entre a estimativa pessimista e otimista ($V = ((P\text{-}O)/6)^2$) e o desvio padrão ($DV = (P\text{-}O)/6$), que mostra o quanto a duração da atividade pode variar para mais ou para menos dentro dessa estimativa. Costumo somar o desvio padrão ao resultado do PERT, para ter um respiro ainda maior sem ficar muito próximo à estimativa pessimista.

. . .

Considerações Finais

Existem inúmeras técnicas de estimativas. Não existe a melhor, existe a estratégia que pode ser mais adequada à você, ao seu time e ao projeto/produto em que você está trabalhando. Neste artigo compartilhei algumas técnicas que venho utilizado ao longo dos anos e têm se mostrado efetivas, sendo meus “2 centavos” para contribuir com esse tema que é tão vasto e polêmico.

Mas as técnicas não ajudam se não houver uma gestão eficiente e se o projeto já começar errado sem elencar adequadamente os requisitos, como pode ser percebido pelos resultados da pesquisa apresentadas na Parte 1. Metodologias, processos e ferramentas não salvam projeto/produto mal gerenciado. Devs: cobrem levantamento de requisitos, conversem com partes interessadas e usuários e se envolvam nesse processo. Gestores: não levem projetos adiante por pressão se não souberem minimamente o que precisa ser feito. Requisitos podem sim

ser elicitados de forma iterativa, mas isso não significa começar a desenvolver a partir de uma página em branco.

. . .

Tem mais alguma sugestão, dica, complemento ao post, falei bobagem? Comentários (respeitosos) são muito bem-vindos!