Estimativas ágeis

Estimativas

- Desde o momento que se faz o comprimisso em desenvolver um *software* tem-se a pergunta: "Quando será terminado?"
- Até mesmo produtos de *software* desenvolvidos continuamente por diversos anos precisam de datas.

Estimativas

- O simples fato de supor quando um *software*, ou parte dele, será entregue já é uma estimativa.
- Estimar nada mais é que uma previsão do futuro.
- Na quase totalidade das vezes, a previsão não será precisa.

Estimativas

- O fato de ser impreciso não torna estimar inútil ou sem ciência alguma.
- Pelo contrário, todo grupo ou organização deve continuamente melhorar sua maneira de estimar, tanto utilizando experiências anteriores quanto aprimorando técnicas.

Estimativas

- Apesar da literatura em Engenharia de *Software* se preocupar bastante com métodos formais para estimativa, chegando até a vislumbrar o uso de Inteligência Artificial para fazê-lo, o que se tem são formas pouco estruturadas e que dependem diretamente da experiência dos envolvidos.
- Nos métodos ágeis isso não é diferente.
- O foco dos métodos ágeis nas conversas entre os envolvidos, no planejamento incremental e em estar sempre pronto para mudar os rumos implica muitas vezes em colocar em segundo plano certos formalismos.

Story points

- É importante que a equipe de desenvolvimento seja a responsável por estimar os requisitos coletivamente.
- Moløkken-Østvold e Jørgensen (2004) mostram que grupos com quatro *experts* tendem a fazer estimativas menos otimistas do que um único indivíduo.
- Um trabalho anterior de Jørgensen já havia mostrado que quando a estimativa é feita pelos desenvolvedores tem mais acurácia que por equipe externa.
- Mais do que isso, pessoas com grande interesse na realização do projeto tendem a fornecer estimativas irreais para que os projetos sejam aprovados.

Story points

Alsaadi e Saeedi (2022) colocam como desafios para estimar os requisitos[1]:

- inexperiência dos membros da equipe;
- pressão externa;
- influência da sequência de estimativa das *stories*.

Na literatura há trabalhos (Grimstad e Jørgensen, 2009; Jørgensen e Halkjelsvik, 2020) que indicam que após estimar *stories* pequenas, a equipe tende a subestimar as demais *stories* e após estimar *stories* grandes, a superestimar as demais.

[1] mais especificamente, *user stories*

Story points

- As *user stories* são (ou deveriam ser) um monopólio para a análise de requisitos dentro dos métodos ágeis.
- Porém, tanto a natureza das *user stories* quanto dos próprios métodos ágeis (iterativos incrementais) criam dificuldades para estimativas e planejamentos a longo prazo.

Story points

- O fato dos cartões serem simples lembretes e não definirem exatamente os requisitos, podem tornar, em um primeiro momento, nebulosas as formas de estimativa.
- O contexto, porém, faz toda diferença.
- Ao mesmo passo que perde-se uma dada "segurança"em estimar em relação ao que é feito nos métodos orientados a planos, ganha-se em reatividade a mudanças.
- De nada adianta ter a melhor das estimativas se as mudanças são inevitáveis.
- Pelo contrário, planejar e estimar consome muitos recursos.

Story points

- De nada adianta ter a melhor das estimativas se as mudanças são inevitáveis.
- Pelo contrário, planejar e estimar consome muitos recursos.

Story points

- O fato de tornar mais "nebulosas"as estimativas não quer dizer que os métodos ágeis ou as *user stories* tornem as estimativas mais imprecisas.
- O que ocorrer é que o método de estimar deve mudar.
- Seria contraditório utilizar um paradigma para levantar, analisar e implementar os requisitos ao mesmo tempo que se usa uma forma de estimar mais arcaica.

Story points

- Os métodos ágeis e as *user stories* trazem a estimativa por *story points*.
- *Story points* são uma medida relativa.
- Na literatura, é possível ver os *story points* representando tamanho (complexidade) ou esforço.
- Independente do que significarem os *story points* para uma equipe, eles devem ser claros e estáveis.

Story points

Tem-se exemplo de duas *user stories* e suas estimativas em *story points* para um sistema de *marketplace*:

- Eu, enquanto Cliente, desejo pagar minhas compras com cartão de crédito. **SP: 3**
- Eu, enquanto Logista, desejo cadastrar produtos para poder vendê-los. **SP: 2**

Story points

- O ponto central aqui é que deve haver uma triangulação entre as *user stories*.
- As *user stories* com mais pontos atribuídos devem ser as mais complexas (ou requiram mais esforço).
- A equipe deve garantir que seja mantida a devida proporção entre as *user stories*.

Story points

- O ponto central aqui é que deve haver uma triangulação entre as *user stories*, buscando que as *user stories* com mais pontos atribuídos sejam mais complexas (ou requiram mais esforço).
- As faixas de pontuações variam de equipe para equipe, sendo a sequência de Fibonacci bastante utilizada.
- Deve-se tomar cuidado com valores muito altos porque podem significar *stories* muito grandes e que deverão ser divididas.

Story points

- Entre complexidade e esforço há uma dicotomia que deve ser bem discutida entre a equipe.
- A complexidade refere-se ao quão difícil é implementar um dado requisito.
- Já esforço é o quanto de trabalho será exigido da equipe para implementar esse mesmo requisito.

Story points

- Os defensores da estimativa por complexidade usarão o argumento que o esforço para implementar algo dependerá da experiência da equipe.
- Isso faz sentido se observado que nos métodos ágeis o código é coletivo e que todos os desenvolvedores são iguais perante ao método.
- Além disso, os métodos ágeis incentivam desenvolvedores "polivalentes".

Story points

- O incentivo à "polivalência"é um ponto positivo dos métodos ágeis.
- Se os membros da equipe de desenvolvimentos são mais "completos"em seus conhecimentos, diminui-se o impacto do *turnover*[2]

[2] *Turnover é quando a empresa/equipe perde um funcionário/membro.*

Story points

Esta "polivalência" tem como pontos fortes:

- menor dependência de membros individuais da equipe;
- mais capacidade de criar soluções;
- maior tendência à equipe aprender novas tecnologias e atualizar-se constantemente.

Story points

- Estimar baseado em complexidade pode aumentar o erro na estimativa.
- Mike Cohn dá como exemplo, no seu artigo "*Story Points Estimate Effort Not Just Complexity*", um projeto onde a equipe é formada por um cirurgião neurologista e uma criança.
- O projeto possui dois requisitos, onde um é uma cirurgia neurológica relativamente simples[3] e colar X selos de carta.
- O primeiro é muito mais complexo que o segundo.
- Se a criança ficar responsável pela cirurgia a medida por complexidade poderá até estar de acordo com a realidade.
- Porém, se o número de selos for grande demais, mesmo que o cirurgião realize a cirurgia e a criança cole os selos, o tempo poderá ser o mesmo, criando um erro de estimativa.

[3] Se é que isso é possível.

Story points

Entretanto, Mike Cohn esquece é que estimar por esforço possui alguns problemas:

- se a *user story* é tão gigantesca e repetitiva como colar X selos, talvez ela deva ser quebrada em várias partes ou até mesmo reescrita;
- acopla cada *story point* a uma quantidade de tempo;
- acopla cada *user story* a um pequeno grupo de desenvolvedores dentro da equipe[4].
- se à "pessoa certa"é dada a atividade, então não há sentido em estimar coletivamente os requisitos, pois se um é a "pessoa certa", então ele também tem a *expertise* para estimar o requisito.

[4]Quando não a um único indivíduo.

Story points

- Mike Cohn, porém, está correto em dizer que a complexidade não é o bastante para estimar um requisito.
- Fatores como risco e incerteza também devem ser levados em conta.

Planning Poker™

- O Planning Poker™é uma técnica utilizada pelas equipes ágeis para a estimativa das *user stories*.
- Ela é derivada da técnica Wideband Delphi.

Planning Poker™

- No Planning Poker™, cada desenvolvedor recebe um baralho de cartas.
- Cada carta possui um valor representando uma quantidade de *story points*.
- Uma *user story* é lida em voz alta e os desenvolvedores podem fazer perguntas para esclarecer detalhes.
- Sanadas as dúvidas, todos os desenvolvedores escolhem, cada um, uma carta e a mostram a todos simultaneamente.
- Caso não haja consenso imediato (todos colocarem o mesmo valor), os desenvolvedores que usaram a maior e a menor carta argumentam o porquê o fizeram.
- O processo repete-se até que haja consenso.

Velocidade

- Ao se usar *story points* para estimar os requisitos, muda-se também a maneira como se vê a eficiência da equipe de desenvolvimento.
- Como as iterações possuem tempo fixo, a quantidade de *story points* realizadas pela equipe em uma iteração é chamada **velocidade**.

Velocidade

- Se uma equipe completa uma quantidade de *user stories* que tenha suas complexidades somadas igual a 30 *story points*, então a velocidade do time é igual a 30.
- Para a próxima iteração espera-se a equipe consiga repetir a mesma velocidade.

Velocidade

São fatores que podem afetar a velocidade da equipe:

- *expertise* do domínio pela equipe;
- domínio das tecnologias pela equipe;
- coesão da equipe;
- reuniões;
- feriados;
- etc.

Velocidade

A velocidade leva em conta também que:

- não foram feitas horas extras;
- as estimativas das *user stories* foram coerentes;
- as *user stories* realizadas foram bem escritas e independentes.

Referências

- Cohn, Mike. User Stories Applied: For Agile Software Development. 1ed. 2004. Addison Wesley.
- Cohn, Mike. Four Reasons Agile Teams Estimate Product Backlog Items. 2022. https://www.mountaingoatsoftware.com/blog/four-reasons-agile-teams-estimate-product-backlog-items
- Cohn, Mike. Story Points Estimate Effort Not Just Complexity. 2010. https://www.mountaingoatsoftware.com/blog/its-effort-not-complexity
- Cohn, Mike. Agile Estimating and Planning. 1ed. 2006. Prentice Hall.
- Alsaadi, Bashaer; Saeedi, Kawther. Data-driven Effort Estimation Techniques of Agile User Stories: a systematic literature review. 2022. Artificial Intelligence Review.
- Moløkken-Østvold, Kjetil; Jørgensen, Magne. Group Processes in Software Effort Estimation. 2004. Empirical Software Engineering.
- Sutherland, Jeff. Story Points: Why are they better than hours?