Influência da IC no tempo de revisão

O tempo total de revisão do código aumenta de forma proporcional ao tempo total de execução da IC.



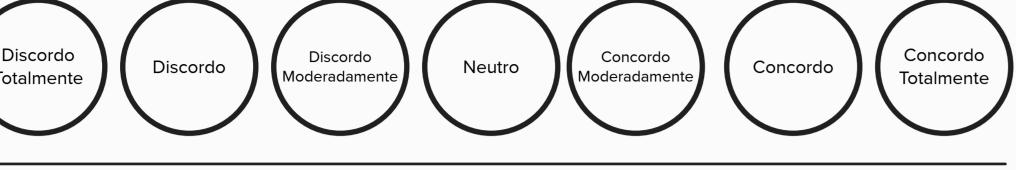
Pontos contrários Pontos a favor

Consigo enxergar cenários em que a revisão do código seria simples, mesmo que a pipelina seja demorada. Por outro lado, se o passo de revisão for dependendo de passos da IC aí sim, eu concordaria.

Dependendo do processo, pode ser que o revisor espere build e relatórios OK gerados na IC para iniciar a revisão.

Influência da IC no tempo de revisão

Quanto mais pipelines de IC são executados mais o tempo de revisão aumenta.



Pontos contrários

O processo pode levar

à revisão apenas de

issues assinaladas

como concluídas pelo

desenvolvedor,

independente dos

pipelines envolvidos.

Se o processo de oipeline demorar, fica a dúvida se compensa

Pontos a favor

revisar o código antes (visto que o código pode não passar nos testes do pipeline)

Efeitos das más práticas de IC na revisão de código

Quais más práticas de IC podem afetar a revisão de código?

5 - A falta de testes de regressão afeta na garantia da qualidade do software, visto que a integração de novas ıncionalidades podem afetar no que já estava funcionando antes.

B - Discordo. Acredito

que o desenvolvedor

certo controle do

ambiente, mas não

total do ambiente.

6 - Passos manuais

são passos feitos

para serem

esquecidos. Além

disso, a integração

fica parcialmente

contínua.

10 - É uma falha

comum manter as

configurações

padrões que nem

sempre atendem as

demandas do time.

recisaria ter o controle

precisa de fato ter um

? - Branches divergentes são especialmente complicados quando os erges acontecem e equipe ão possui experiência para resolver os merges adequadamente. Idealmente, os merges não deveriam gerar conflitos.

- Concordo que apenas o

desenvolvedores que

optaram por receber as

otificações não seja o ideal

No entanto, acho que essas

notificações podem ser

enviadas somente para

partes interessadas, uma

quipe específica, um grupo

de desenvolvedores.

' - Outra má prática

lássica. Um ambiente

de testes similar ao

de produção evita

que alguns

roblemas ocorram

somente em

produção.

12 - Quando o

script impede a

continuidade do

prática se torna

ainda mais graves.

2 - conflitos de 2 - Branches divergentes acontece muito com a demora demandam los desenvolvedore atualizarem a suas tempo para branches com a

padrão.

5 - Uma

pipeline sem

testes é um

pipeline

A má prática

9 é bem

parecida

13 e 10 são

bem

estar

com a 7

resolução. 6 - passos manuais

código

no pipeline poden levar a atrasos, ja que dependem de alguém (humano executar algo.

12 - scripts longo ou com tarefas demoradas odem ser lento e atrasar o

processo.

funcionalidades em vez de usar toggles (interruptores) de funcionalidade. Os toggles permitem que as funcionalidades sejam desenvolvidas e ocultas até estarem prontas para serem ativadas, enquanto os ramos de funcionalidade podem levar a problemas de integração e dificuldades na entrega contínua.

Esta prática envolve a criação de ramos separados (feature branches) para desenvolver novas

2. Divergent Branches:

Isso ocorre quando os ramos do código se afastam demais uns dos outros e se tornam difíceis de serem mesclados novamente. Divergent branches podem causar conflitos de mesclagem significativos e tornar a manutenção do código mais difícil.

3. Developers do not have complete control of the environment:

1. Feature branches are used instead of feature toggles:

Isso significa que os desenvolvedores não têm controle total sobre o ambiente em que estão trabalhando, o que pode levar a problemas de compatibilidade e dificultar a reprodução de problemas em ambientes de desenvolvimento.

4. Failures notifications are only sent to teams/developers that explicitly subscribed:

Se apenas as equipes ou desenvolvedores que se inscreveram para receber notificações são informados sobre falhas na integração contínua, isso pode resultar em atrasos na resolução de problemas, já que outros envolvidos podem não estar cientes dos problemas.

5. Missing tests on feature branches:

Desenvolver funcionalidades em ramos separados sem escrever testes adequados pode levar a problemas de qualidade, já que não há garantia de que as novas funcionalidades funcionarão corretamente e não afetarão negativamente o código existente.

6. Some pipeline's tasks are started manually:

Tarefas que são iniciadas manualmente em um pipeline de integração contínua podem levar a inconsistências e atrasos, além de aumentar o risco de erros humanos.

7. Lack of testing in a production-like environment:

Não realizar testes em um ambiente semelhante ao de produção pode resultar em problemas que não são detectados até que a funcionalidade seja implantada no ambiente de produção real.

8. Test cases are not organized in folders based on their purposes:

A organização inadequada dos casos de teste pode tornar difícil para os desenvolvedores e testadores encontrar e gerenciar os testes necessários, tornando o processo de teste menos eficiente

9. The CI server hardware is used for different purposes other than running the CI framework: Usar o hardware do servidor de integração contínua para outros fins pode afetar o desempenho e a confiabilidade do processo de integração contínua.

10. External tools are used with their default configurations:

Usar ferramentas externas com configurações padrão pode não atender às necessidades específicas do projeto, resultando em subutilização das ferramentas e funcionalidades não aproveitadas.

11. A build fails because of some flakiness in the execution, whereas it should not:

Flakiness refere-se a instabilidade nos testes ou na construção do código. Falhas intermitentes podem ser frustrantes e difíceis de depurar, prejudicando a confiabilidade do processo de integração contínua.

12. Lengthy build scripts:

Scripts de construção longos podem tornar o processo de compilação demorado e complexo, afetando o tempo de entrega e a manutenção.

13. Quality gates are defined without developers considering only what is dictated by the customer: Estabelecer critérios de qualidade sem levar em consideração os requisitos de todas as partes envolvidas pode resultar em processos de integração contínua que não atendem às expectativas do cliente ou que são excessivamente rigorosos.

14. Developers and operators are kept as separate roles:

Manter desenvolvedores e operadores como papéis separados pode levar a lacunas de comunicação e atrasos na resolução de problemas, pois os desenvolvedores podem não ter uma compreensão completa das operações em produção e vice-versa.

Benefícios e desafios da IC na revisão de código

Desafios Benefícios

Garantir a qualidade de código é desafiador ndependentemente de pipeline. Uma pipeline bem sucedida ajuda, mas a revisão por si só é um desafio.

Implementar

diferentes tipos de

especialmente

quando lidam com

bancos de dados n

A adequação a

qualquer

processo e

sempre um

risco.

IC é bem desafiador.

testes,

Conflitos no ódigo poder atrasar bastante d processo.

O processo

usado afeta IC e

Adequação de

processo é

desafiador.

de sinalização mais rápida, visto de problema que os testes se tornam mais facilitam a frequentes e automatizados.

automatização da implantação de novas versões

Ferramentas

odem ser antecipado m passos da pipeline O que pode ajudar na iorização de tarefas melhor gerenciament de tempo.

Alguns problemas

Ajuda a encontrar

Facilita na Passos que envolvem a ecagem de seguranç e vulnerabilidade sã novas ssenciais e podem se atualizações feitos de forma produto para cliente.

tomática. Dificilment ais problemas seriam ncontrados na revisão.

Uma boa IC simplifica a realização de tarefas repetitiva por meio de