Integração Contínua

Problema geral.

Encontrar uma boa solução para um problema existente e entregar esta solução para o cliente com o menor tempo possível.

Foco da abordagem.

Unir e coordenar todas as atividades do processo de desenvolvimento do software a fim de torná-las mais eficientes e confiáveis.

Fazer com que a equipe trabalhe junto e coopere durante o processo, de maneira eficiente.

Obstáculos a serem superados.

Implantar o software manualmente: Obtemos este problema quando olhamos para cada passo no processo de implantação do software de forma isolada, atômica.

-Testes manuais para confirmar funcionamento da aplicação.

-Correção no processo de entrega de uma versão no dia em que o mesmo está sendo executado.

Solução: Automatizar ao máximo o processo de implantação.

Obstáculos a serem superados.

Implantar em um ambiente similar ao de produção somente quando o desenvolvimento estiver completo: quando o ambiente em que a aplicação vai ser implantada é diferente do esperado.

-Há pouca ou nenhuma colaboração entre o time de desenvolvimento e as pessoas que fazem a implantação do software.

Solução: integrar todas as atividades de teste, implantação e entrega de versão ao processo de desenvolvimento.

Obstáculos a serem superados.

Gerência de configuração manual dos ambientes de produção: Gerenciar a configuração dos seus ambientes de produção por meio de uma equipe de operação.

-A equipe de operações demora muito para preparar o ambiente para entrega de uma versão.

Solução: implantar um sistema de controle de versão para automatizar o processo.

Por que automatizar o processo de entrega?

Se o processo de entrega não é automatizado, ele não é passível de repetição. Ou seja cada processo se torna um processo único, logo é uma tarefa muito mais complexa de se analisar para buscar uma futura otimização.

Por que utilizar-se de entregas frequentes?

Quando possuímos entregas de maneira mais frequente, diminuímos a discrepância entre as versões, reduzindo assim os riscos recorrentes ao funcionamento do software e facilitando assim o processo de reversão quando encontrado um erro. Além de possibilitar um feedback mais rápido e constante.

Por que devemos utilizar um controle de versão?

Todo artefato utilizado durante a produção e entrega do software deve ser mantido em algum tipo de sistema de versionamento, e isto deve ser classificado de acordo com seu nível de importância.

Gerência de configuração.

Gerência de configuração se refere ao processo pelo qual todos os artefatos relevantes ao seu projeto, e as relações entre eles, são armazenados, recuperados, modificados e identificados de maneira única.

Extreme programming (1996)

- Refatoração
- TDD
- Propriedade coletiva de código
- Design incremental
- Programação em pares
- Padrões de códigos
- **Integração contínua**

Manifesto ágil (2001)

- **Indivíduos e interações** mais que processos e ferramentas
- **Software em funcionamento** mais que documentação abrangente
- **Colaboração** com o cliente mais que negociação de contratos
- **Responder a mudanças** mais que seguir um plano

DevOps (2009)

DevOps é o alinhamento do time de desenvolvimento com o time de operações, em relação à processos, ferramentas e responsabilidades, visando acelerar as entregas em produção com um elevado grau de qualidade.

Integração Contínua

Prática que encoraja desenvolvedores a integrar seu trabalho frequentemente para que possíveis problemas sejam detectados e corrigidos rapidamente.

Infra estrutura necessária para implementar

- Controle de versão
- Processo **automatizado** de compilação
- Aceitação da equipe

Pipeline de Implantação

Pipeline de Implantação prático

Pipeline de Implantação prático

No mínimo uma vez por dia, se não for possível então que pelo menos atualize a cópia local

Software de integração contínua

Automatizam o processo de build após cada commit no repositório

Name	Platform	License	Builders: Windows	Builders: Java	Builders: other	Notification	Integration, IDEs	Integration, other
AnthillPro	Cross-platform	Proprietary	MSBuild, NAnt, Visual Studio	Ant, Maven 1-2-3	Shell script, batch script, cross-platform command-line, Groovy, Make, RTC Jazz, TFS Build, Custom Script Interpreter	Email, XMPP, RSS, Systray	Eclipse, Visual Studio	Many
Apache Continuum	JDK, web container	Apache 2.0	Unknown	Maven 1-2-3	Shell script[2]	Mail, XMPP and Google Talk, MSN, IRC, report deployment with wagon	Unknown	Unknown
Apache Gump	Python	Apache 2.0	Unknown	Ant, Maven 1	Unknown	Email	Unknown	Unknown
AppVeyor	Hosted	Proprietary	Visual Studio, MSBuild, Psake	No	Custom Script, PowerShell	Email, HipChat, Slack, Catlight	No	GitHub, Bitbucket, Kiln, Windows Azure
Bamboo	Web container	Proprietary	MSBuild,[3] NAnt,[4] Visual Studio[5]	Ant,[6] Maven 1-2-3[7]	Custom script, command-line tool, Bash, Xcode,[8] Phing,[9] Grunt	XMPP, Google Talk, Email, RSS, Remote API, HipChat	IntelliJ IDEA, Eclipse, Visual Studio	FishEye, Jira, Clover, Bitbucket, GitHub
Buddy	Cross-platform	Proprietary	No	Ant, Maven, Gradle	Elixir, Go, Haskell, Node.js, PHP, Python, Ruby, .NET Core	Desktop, Email, Slack, SMS	No	Amazon Web Services, Bitbucket, GitHub, GitLab, Google Cloud Services, Heroku, Modulus
BuildBot	Python	GPL	Command-line	Command-line	Command-line	Email, Web, GUI, IRC	Unknown	Unknown
BuildMaster	Cross-platform	Proprietary	Yes	Yes	Cross-platform command-line	Email, custom	No	Many
CABIE	LAMP	GPL2	Unknown	Unknown	Unknown	Web	Unknown	Unknown
CruiseControl	Cross-platform	BSD-style	NAnt, Rake, Xcode	Phing, Apache Ant, Maven	catch-all 'exec'	Email, CCTray	Eclipse	Unknown
CruiseControl.NET	Cross-platform	BSD-style	MSBuild, NAnt, Visual Studio	Unknown	Command-line	Email, CCTray, RSS	Unknown	Unknown
GoCD	Cross-platform	Apache 2.0	Command-line	Command-line	Command-line	Email, hipchat, Slack, Gerrit, Gitter, Riemann etc	No	GitHub
Jenkins-Hudson	Web container	Creative Commons and MIT	MSBuild, NAnt	Ant, Maven 2, Kundo	Cmake, Gant, Gradle, Grails, Phing, Rake, Ruby, SCons, Python, shell script, command-line	Android, Email, Google Calendar, IRC, XMPP, RSS, Twitter, Slack, Catlight, CCMenu, CCTray	Eclipse, IntelliJ IDEA, NetBeans	Bugzilla, Google Code, Jira, Bitbucket, Redmine, FindBugs, Checkstyle, PMD and Mantis, Trac, HP ALM
OpenMake Software Meister	Cross-	Proprietary	MSBuild, NAnt, Visual	Ant, Maven 1-2-3	Shell script, batch script, cross-platform command-line, Groovy, Make,	Email, XMPP, RSS, Systray	Eclipse, Visual Studio	Bugzilla, Google Code, Jira, Bitbucket, Redmine, FindBugs,

Buddy	platform	Proprietary	No	Ant, Maven, Gradle	Python, Ruby, .NET Core	Desktop, Email, Slack, SMS	No	GitHub, GitLab, Google Cloud Services, Heroku, Modulus
BuildBot	Python	GPL	Command-line	Command-line	Command-line	Email, Web, GUI, IRC	Unknown	Unknown
BuildMaster	Cross-platform	Proprietary	Yes	Yes	Cross-platform command-line	Email, custom	No	Many
CABIE	LAMP	GPL2	Unknown	Unknown	Unknown	Web	Unknown	Unknown
CruiseControl	Cross-platform	BSD-style	NAnt, Rake, Xcode	Phing, Apache Ant, Maven	catch-all 'exec'	Email, CCTray	Eclipse	Unknown
CruiseControl.NET	Cross-platform	BSD-style	MSBuild, NAnt, Visual Studio	Unknown	Command-line	Email, CCTray, RSS	Unknown	Unknown
GoCD	Cross-platform	Apache 2.0	Command-line	Command-line	Command-line	Email, hipchat, Slack, Gerrit, Gitter, Riemann etc	No	GitHub
Jenkins-Hudson	Web container	Creative Commons and MIT	MSBuild, NAnt	Ant, Maven 2, Kundo	Cmake, Gant, Gradle, Grails, Phing, Rake, Ruby, SCons, Python, shell script, command-line	Android, Email, Google Calendar, IRC, XMPP, RSS, Twitter, Slack, Catlight, CCMenu, CCTray	Eclipse, IntelliJ IDEA, NetBeans	Bugzilla, Google Code, Jira, Bitbucket, Redmine, FindBugs, Checkstyle, PMD and Mantis, Trac, HP ALM
OpenMake Software Meister	Cross-platform	Proprietary	MSBuild, NAnt, Visual Studio	Ant, Maven 1-2-3	Shell script, batch script, cross-platform command-line, Groovy, Make, RTC Jazz, TFS Build, Custom Script Interpreter	Email, XMPP, RSS, Systray	Eclipse, Visual Studio	Bugzilla, Google Code, Jira, Bitbucket, Redmine, FindBugs, Checkstyle, PMD and Mantis, Trac
Travis CI	Hosted	MIT	No	Ant, Maven, Gradle[10]	C, C++, Clojure, Elixir, Erlang, Go, Groovy, Haskell, Java, Node.js, Perl, PHP, Python, Ruby, Rust, Scala, Smalltalk	Email, Campfire, HipChat, IRC, Slack, Catlight, CCMenu, CCTray	No	GitHub, Heroku
TeamCity	Web container	Proprietary	MSBuild, NAnt, Visual Studio, ReSharper-based .NET code analysis	Ant, Maven 2-3, Gradle, IntelliJ IDEA-based build and code analysis	command-line, PowerShell,[11] Xcode,[12] Rake, FxCop	Email, XMPP, RSS, IDE, SysTray, Catlight	Eclipse, Visual Studio, IntelliJ IDEA, RubyMine, PyCharm, PhpStorm, WebStorm	JetBrains YouTrack, Jira, Bugzilla, FishEye, FindBugs, PMD, dotCover, NCover
Team Foundation Server, Visual Studio Team Services (VSTS)	Cross-platform	Proprietary, MIT	MSBuild, Visual Studio	Ant, Maven, Gradle, Android	C, C++, Go, Groovy, Java, Node.js, Perl, PHP, Python, Ruby	Email, SOAP, Catlight	Visual Studio, Eclipse, IntelliJ IDEA, Android Studio, Visual Studio Code	GitHub, Jenkins, Slack, Hipchat, FindBugs, Checkstyle, PMD
Vexor	Hosted	Proprietary	No	Unknown	Ruby, Clojure, Scala, Python, Node.js, Go, Rust, Haskell	Email, HipChat, Slack	Unknown	GitHub, Bitbucket, Gitlab

Requisitos para a Integração Contínua

1. Controle de versão;
2. Commits frequentes
3. Testes automatizados.

Controle de Versão

1. Trabalho em equipe;
2. Controle do histórico;
3. Ramificações (branches).

Check-ins

Check-ins(commits) regulares para o **trunk**.

Menor risco no desenvolvimento.

Teste de Software

Forma de procurar garantir que o produto de software atende aos requisitos que foram definidos para ele.

“o processo de execução de um programa com a intenção de encontrar erros.” (Myers, 2004)

Tipos de teste

1. Teste Funcional (Obs: nome não relacionado aos requisitos funcionais);
2. Teste Estrutural;
3. Teste baseado em defeito.

Fases de teste

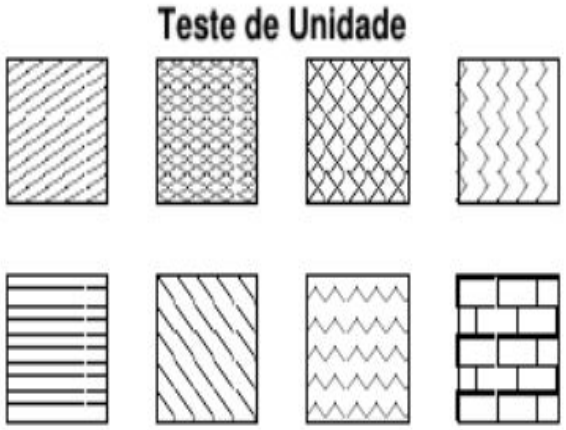
Delamaro (2016) e Baresi (2006) definem as seguintes fases de testes:

1. Teste Unitário;
2. Teste de Integração;
3. Teste de Sistema;
4. Teste de Aceitação;
5. Teste de Regressão.

Teste de Unidade

Teste Procedimental

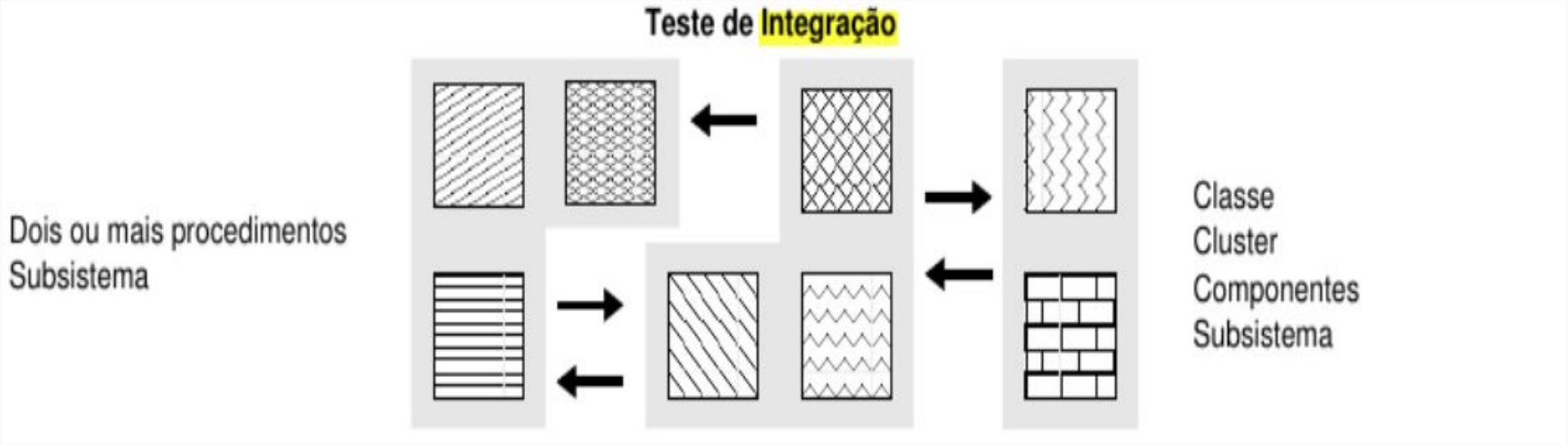
Procedimento ou Sub-rotina



Teste Orientado a Objeto

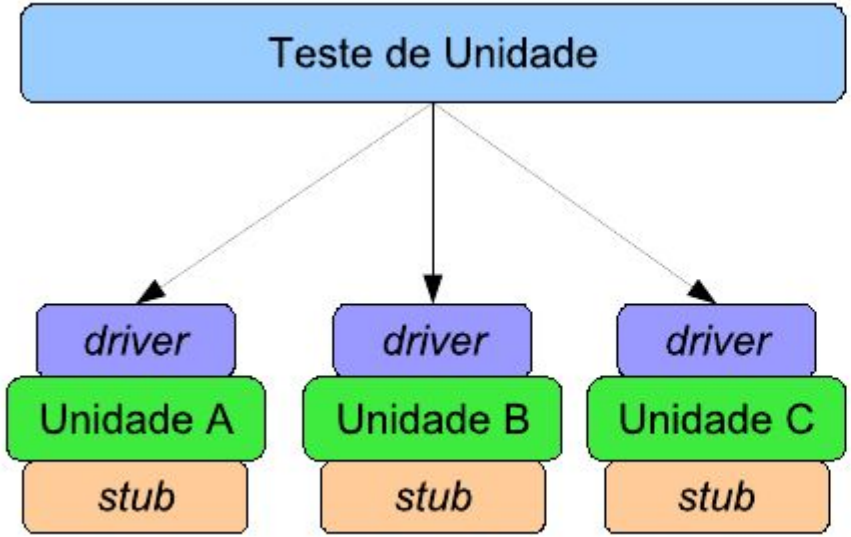
Método

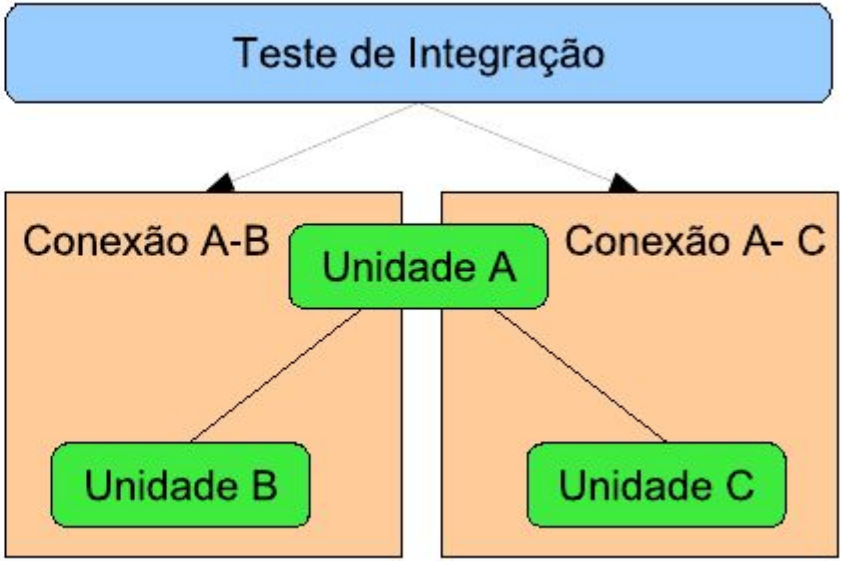
Teste de Integração



Teste de Sistema e Aceitação

Teste de Unidade e Teste de Integração





Teste de Sistema

Teste de integração de mais alto nível.

SO, banco de dados, hardware, manual do usuário, treinamento, etc.

Teste de Aceitação

Verificar se o programa desenvolvido atende às exigências do cliente.

Teste de Regressão

Verifica se nenhum efeito colateral foi introduzido.

Na IC, os testes unitários e de integração já escritos são também caracterizados como testes de regressão, já que são executados a cada commit.

Tempo de execução

Testes são executados a cada commit.

Duração vs Cobertura.

Integração Contínua no GitHub

Fork: Produção -> Fork

Clone: Fork -> Local

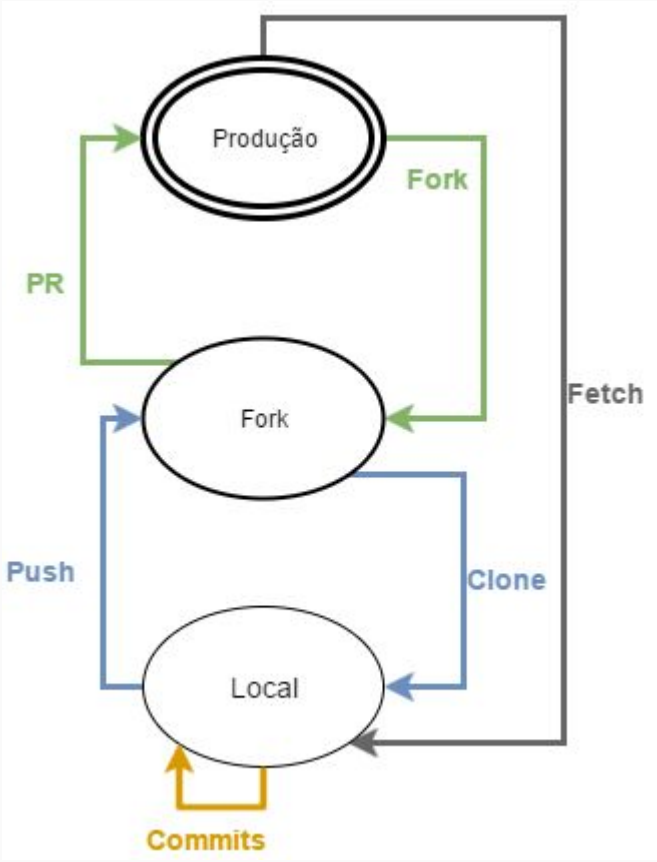
Commits: Local -> Local

Push: Local -> Fork

PR (Pull Request): Fork -> Produção

Fetch: Produção -> Local

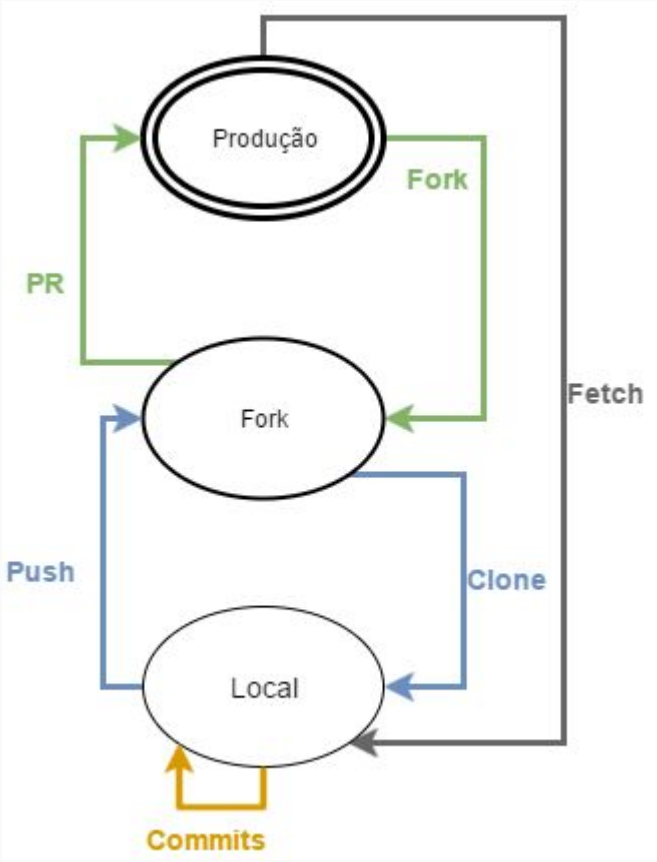
- Branches
- Ferramentas de automação



Princípios da Integração Contínua

- Não introduza código novo se os testes estão falhando
  - Código novo dificulta a solução do problema existente
  - Testes não garantem a validade do código novo

- Atualize sua cópia local com o ambiente de produção (fetch) e rode testes antes de um PR
  - Garante que suas mudanças funcionam com a versão mais atual do código de produção



Princípios da Integração Contínua

- Espere o sucesso dos testes para começar uma nova tarefa
- Não comente testes que estão falhando
- Nunca pare de trabalhar com o código de produção quebrado
- Limite de tempo para correção de problemas

Princípios da Integração Contínua

- Assuma responsabilidade pelas quebras causadas por suas mudanças
- Esteja sempre preparado para voltar à revisão anterior

Práticas da Integração Contínua

- eXtreme Programming
  - Princípio de XP
  - Integração Contínua x Entrega Incremental
    - Releases

- Testes unitários lentos que funcionam não funcionam

Práticas da Integração Contínua

- Falhas arquiteturais causam erros de aceitação
  - Regras de arquitetura e estrutura da aplicação devem ser cobertas em testes

  Javascript Isomórfico (Isomorphic Javascript)

    - google-cloud - apenas NodeJS
    - document - apenas browser

Práticas da Integração Contínua

- Divergência de estilo de código causa erro de aceitação
  - Linters

- Go
  - go fmt
- Javascript
  - Typescript

Test Driven Development (TDD)

- Testes são escritos antes do código a ser testado
- Red, Green, Refactor
  - Escreva um teste e o execute. É essencial que ele **falhe**.
  - No código, faça a o que for preciso para que o teste **passe**.
  - **Refatore** o código, melhorando legibilidade e adequando aos padrões do projeto.
- Se o código existe, então está testado.

Test Driven Development (TDD)

- YAGNI (You ain’t gonna need it)
- KISS (Keep it simple stupid)
- Aprender o sistema estudando resultados de testes

“...única maneira eficaz de obter cobertura excelente é através de TDD[...] é essencial para que seja alcançada a prática de entrega contínua”

Jez Humble, Estrutura de Dados

Test Driven Development (TDD)

“Think-first rather than test-first is the way to go.” - Ian Sommerville

“TDD is dead. Long live testing.” - David Hansson (Ruby on Rails)

“Is TDD Dead?” - Martin Fowler, Kent Beck (XP), David Hansson

Conclusão

Como desfecho da dinâmica, apresentaremos a conclusão de maneira geral para todos.

Referências

- HUMBLE, J.; FARLEY, D. **Entrega contínua**: como entregar software. Porto Alegre: Bookman, 2013. 496p.
- DELAMARO, Márcio Eduardo; MALDONADO, José Carlos - **Introdução ao Teste de Software** - 2 Ed. Elsevier - Campus, 2016.
- BINDER, R. V. **Testing object-oriented systems: Models, patterns and tools**, v. 1. Addison Wesley Longman, Inc., 1999.
- MYERS, G. **The art of software testing.** John Wiley and Sons, 2004.
- BARESI, L., PEZZÈ, M. **An Introduction to Software Testing**. Electronic Notes in Theoretical Computer Science, Vol. 148, No. 1, pp. 89-111, 2006
- NEVES, Vânia de Oliveira. ***Automatização do teste estrutural de software de veículos autônomos para apoio ao teste de campo***. São Carlos : Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2015. Tese de Doutorado em Ciências de Computação e Matemática Computacional.
- O que significa integração contínua?, Disponivel em <https://aws.amazon.com/pt/devops/continuous-integration/> Acesso 15 maio 2017

Referências

- Sato, D. **DevOps na prática** entrega de software confiável e automatizada, São Paulo, 2013. 262p.
- **Comparison of continuous integration software**, Disponivel em <https://en.wikipedia.org/wiki/Comparison_of_continuous_integration_software/> Acesso 15 maio 2017
- **O que é DevOps?** <https://www.ibm.com/developerworks/community/blogs/rationalbrasil/entry/o_que_devops?lang=en/> Acesso 15 Maio 2017
- **Guidelines for Test-Driven Development** <https://msdn.microsoft.com/en-us/library/aa730844(v=vs.80).aspx>
- **Giving up on test-first development** <http://iansommerville.com/systems-software-and-technology/giving-up-on-test-first-development/> Acesso 15 maio 2017
- **TDD is dead. Long live testing.** <http://david.heinemeierhansson.com/2014/tdd-is-dead-long-live-testing.html> Acesso 15 maio 2017
- **Is TDD dead?** <https://martinfowler.com/articles/is-tdd-dead/> Acesso 15 maio 2017