Técnicas de programação para Games
Aula01
Metodologias Ágeis

Professor Mestre: Adilson Lopes Khouri

4 de novembro de 2018

Sumário

Apresentação

- Adilson Khouri, jogador de Magic the Gathering, nerd, apaixonado por computação e machine learning!

Figura: Ministrando uma palestra no Peru e trabalhando na Argentina

Formação Acadêmica

- Bacharel em Sistemas de Informação (2011 - USP)
- Mestre em Sistemas de Informação (2016 - USP)
- Doutorando em Sistemas de Informação (cursando - USP)

Experiência Acadêmica

- Um ano de estágio em docência na USP
- Publicações Científicas
- Orientação de iniciação científica
- Disciplina de Técnicas de programação em Games no SENAC

Experiência de Mercado

- Programador na consultoria Arbit (2010-2011)
- Programador Itaú-Unibanco (2011-2013)
- Cientista de dados Sr. PagSeguro (2016 - 2018)
- Cientista de dados Sr. NuvemShop (Atual)
- Professor de Programação - SENAC (Atual)

E os senhores?

- Nome
- Graduação / pós-graduação
- Trabalho
- Qual sua experiência com programação e estruturas de dados?

Expectativas

- Quais expectativas com relação à matéria de: “técnicas de programação para games”?
- O que deve ser evitado?
- (E-Mail: 0800dirso@gmail.com)

Cronograma

Aula	Conteúdo
12/04/2018	XP e banco de dados
17/04/2018	Introdução de estruturas de dados
24/04/2018	Arrays / Matrizes e Ordenação
26/04/2018	Recursão
03/05/2018	Lista Ligada
08/05/2018	Pilha, Fila
10/05/2018	Hash
15/05/2018	Árvore Binária
17/05/2018	Heap
22/05/2018	Grafos
24/05/2018	Prova

Metodologia

- Aulas expositivas teóricas (25%) e práticas (75%)
- Exercícios
- Comparação de algoritmos na classe
- Trabalho em grupo
- Prova individual

Critério Avaliação

- $\frac{Prova + Trabalho}{2}$

Como desenvolver software/Games?

- Podemos sair codificando.... até atingir um objetivo (Code and fix). Basicamente como fazemos exercícios programa da faculdade? Funciona? Há algum problema nessa abordagem?
- Podemos usar alguma metodologia para garantir alguma qualidade nas diversas etapas da construção de um software/game
- Uma metodologia clássica é a: metodologia cascata...

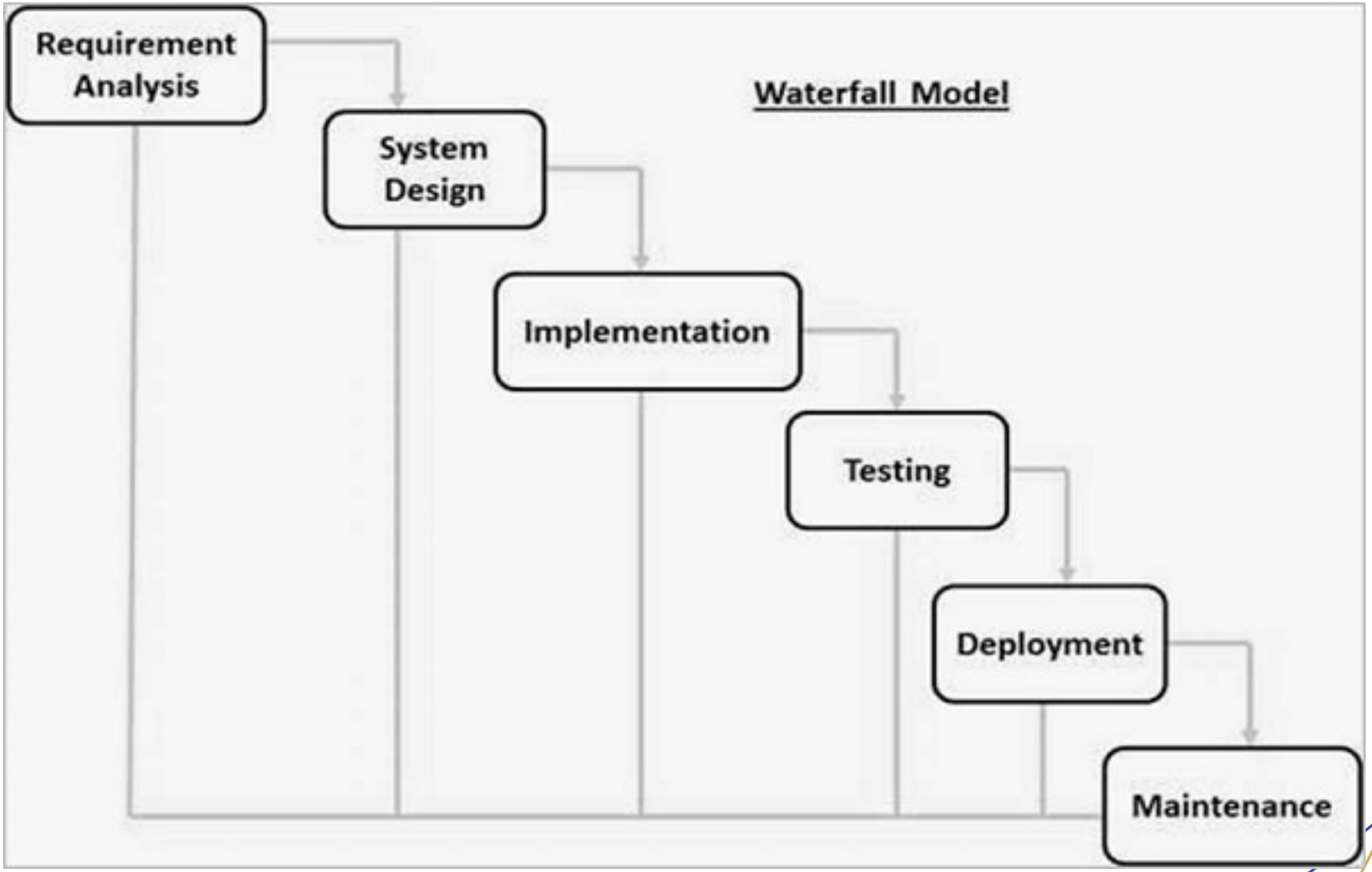

Figura: Etapas da metodologia cascata [1]

Detalhes

- As etapas são bem determinadas, tem começo meio e fim bem definidos e assume os requisitos são claros e bem definidos. Só passamos para a próxima etapa após finalizar com sucesso a etapa anterior. Não podem ocorrer alterações após as etapas
- Começamos por especificar requisitos com o cliente, definir o que deve ser feito, como será feito.
- O passo seguinte exige criar a arquitetura do sistema.
- Após definir a arquitetura temos a codificação.
- Testamos a codificação.
- Entrega (Deploy).
- Manutenção e correção de bugs.

Trade-off

- É simples e de fácil gerenciamento, afinal todas as etapas são rígidas e não permitem alterações.
- É bom para projetos pequenos onde os requisitos são extremamente bem definidos, conhecidos, não mudam e que precisam de uma extensa documentação.
- No mundo real ninguém conhece 100% todos os requisitos no começo do projeto.... Nem mesmo o cliente ou usuário... Muito menos o desenvolvedor...
- Se ocorrer alguma alteração nessa metodologia ocorrerá um caos no projeto...
- Software só é entregue para o cliente/usuário depois que tudo está pronto.

Exercício

- Vamos simular desenvolver um projeto usando a metodologia: “cascata”.
- O jogo que iremos desenvolver será: “Age of Empires”, todos conhecem?
- Vamos escrever no papel, de forma sequencial cada etapa, somente começaremos outra etapa quando acabar a anterior. Devemos descrever/desenhar cada uma das etapas (Sejam criativos)
- Requisitos (O professor fará o papel do cliente)
- Desenho da arquitetura do sistema (O professor fará o papel do gerente)
- Implementação
- Teste
- Deploy
- Manutenção

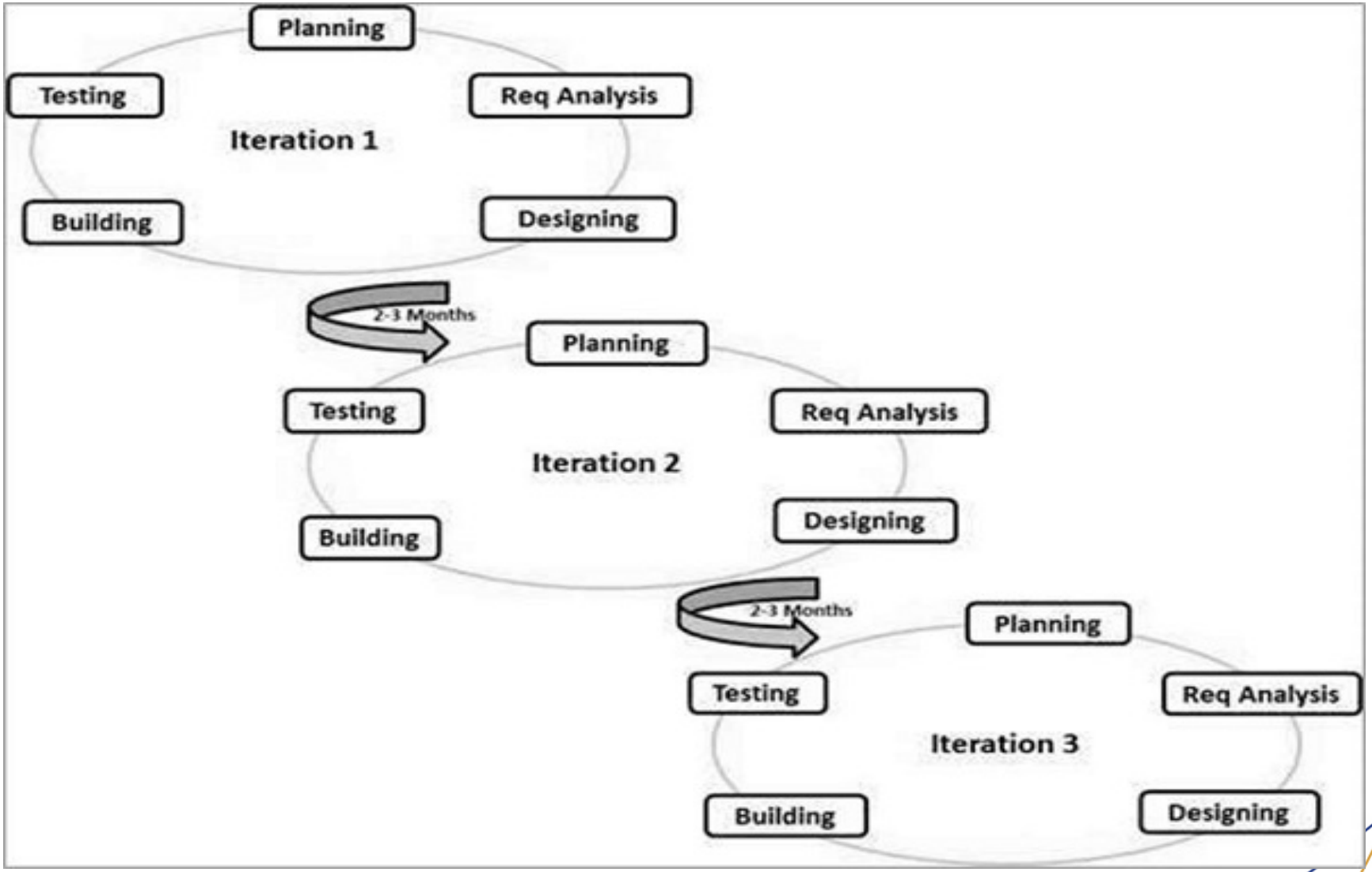

Figura: Etapas da metodologia XP [1]

Detalhes

- Ciclo de iteração curto (15 dias, 1 mês):
- Escrever histórias de usuário
- Escrever os testes de aceitação pelo cliente
- Priorizar as histórias junto com o usuário
- Estabelecer um MVP (produto mínimo que agrega valor para o cliente)
- Codificar
- Testar
- Entregar para o cliente
- Próximo ciclo!
- Não temos a definição perfeita do projeto todo, por isso priorizamos apenas no que conhecemos e sabemos.

Trade-off

- É mais complexo o gerenciamento e compreensão, afinal considera a mudança constante do negócio.
- É realista, considera mudanças, entrega valor rápido, menos propenso ao retrabalho.
- Caso a história não seja usada, o negócio mudou, por exemplo, o tempo gasto nela foi menor que o cascata pois trabalhamos com pequenos ciclos de interação.
- Não existe: “Vamos fazer toda a documentação”, ela é feita apenas para os pontos relevantes da história.
- Programação em pares.
- Cliente presente (de alguma forma mesmo que não fisicamente). Explicar meu caso de cliente ausente na PagSeguro e Itaú.

Roles

- Desenvolvedor – quem irá desenvolver o game
- Coach – é o técnico que garante a aplicação da cultura ágil.
- Testador – quem irá testar o game
- Cliente – Quem informa os desafios de negócio que devemos resolver
- Cleaner – Refatora o código e cobra a equipe toda para manter a limpeza do mesmo
- Tracker – Coleta as métricas do projeto
- Gerente – Cobra as entregas do projeto

Visão

- Time multidisciplinar e coeso
- Time sabe se gerenciar sozinho, gerente é um facilitador
- Time fisicamente próximo
- Reuniões de revisão para avaliar onde podemos melhorar (Não caçar as bruxas!)
- Cliente presente
- Liberação frequente de pequenas entregas (Mas professor... Em games isso é viável?)
- Faça POC e não reaproveite o código da POC! Normalmente é um código inicial para aprendizagem de baixa qualidade

Exercício

- Vamos escrever uma história de usuário, no papel, para desenvolver o jogo: “Age of Empires”. Cada aluno escreverá uma história distinta.
- Exemplo de história de usuário:
- Eu como USUARIO DO SISTEMA gostaria de ter a funcionalidade FUNCIONALIDADE para que me permita OBJETIVO pois isso agrega valor para meu negócio da seguinte forma AGREGAÇÂO DE VALOR

Exercício Priorização

- Agora vamos priorizar as histórias:
- O que agrega mais valor para o cliente primeiro?
- O que queremos garantir que ele já vá testando antes?
- Quais histórias estão incertas?
- O que podemos deixar para depois?
- Quais histórias são imprescindíveis para nós desenvolvermos?

XP x Cascata

- Vamos simular um projeto de game com as duas metodologias e avaliar qual delas é mais útil para a realidade dos games.
- Nosso contexto agora é o jogo: “World of Warcraft”.
- Os alunos serão os programadores e testadores.
- O professor será o cliente e gerente, simulando duas pessoas distintas.

Dúvidas...

Alguma dúvida?

Contato

- E-mail: 0800*dirso@gmail.com* (alunos SENAC)
- E-mail: *adilson.khouri.usp@gmail.com*
- Phone: +55119444 – 26191
- Linkedin
- Lattes
- GitHub

Referências

[1] Tutorialspoint. (2017) Tutorialspoint: simple easy learn. [Online]. Available: https://www.tutorialspoint.com/

[2] A. V. Aho, J. E. Hopcroft, and J. Ullman, *Data Structures and Algorithms*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1983.

[3] K. Beck and C. Andres, *Extreme Programming Explained: Embrace Change (2Nd Edition)*. Addison-Wesley Professional, 2004.

[4] Beck, *Test Driven Development: By Example*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[5] M. Fowler, *Refactoring: Improving the Design of Existing Code*. Boston, MA, USA: Addison-Wesley, 1999.

[6] Geeks. (2018) A computer science portal for geeks. [Online]. Available: https://www.geeksforgeeks.org