



INSTRUÇÕES DO JOGO

Introdução

- Objetivo do jogo: concluir um projeto de software
- Número de jogadores (primeira versão): recomenda-se de 2 a 3 jogadores, mas poderia ser jogado individualmente.
- Composição do jogo: 1 dado, 1 tabuleiro de projeto para cada jogador, 5 cartas de projeto, 10 cartas de equipe, 10 cartas de conceito, 32 cartas de evento, marcadores para cartas de equipe, marcadores de funcionalidades, marcadores de funcionalidades com *bugs* e 2 tabelas de área de jogo.

As Cartas

Carta de Projeto: consiste no objetivo do jogo para cada jogador. Contém:

- Descrição: descrição básica do projeto.
- Número de *releases*: sequência de *releases* a serem completadas e entregues para conclusão do projeto.
- Descrição das *releases*: número de funcionalidades que deve ser produzido para cada componente das *releases*.
- Qualidade: número aceitável de *bugs* que o projeto final poderá conter.
- Verba: quantidade de dinheiro que será arrecadado no início do projeto e ao final de cada *release*.

Carta de Equipe: é utilizada como mão de obra do projeto. Equipes podem ser alocadas dentro do projeto como equipes de desenvolvimento ou equipes de testes a critério do jogador. Quando alocada em desenvolvimento uma equipe irá produzir os componentes das *releases*, efetuando mudanças e corrigindo defeitos. Quando alocada em testes a equipe irá detectar os erros nas *releases* candidatas e reportar às equipes de desenvolvimento. Contém:

- Salário: dinheiro que deve ser pago a equipe ao final de cada turno.
- Adicional por Contratação: taxa que deve ser acrescentada ao valor do salário da equipe cada vez que esta for contratada por um jogador, após a primeira contratação. Deve-se utilizar o marcador de quantidade "I" na carta de equipe para cada adicional de contratação que esta possuir em seu salário.

- Habilidade em Desenvolvimento: número de rolagens de dado permitidas por turno. O somatório dos números tirados nas rolagens de dado da equipe reflete na quantidade de modificações (produção de funcionalidades e correção de defeitos) que a equipe poderá produzir nos componentes do projeto por turno.
- Tendência a *Bugs*: fração que define a quantidade de modificações produzidas com *bugs* pela equipe. Esta fração se refere ao total das modificações dos componentes por turno.
- Habilidade em Testes: número de rolagens de dado permitidas por turno. Cada rolagem de dado deve ser efetuada para somente um componente da *release* a escolha do jogador. O número tirado na rolagem de dado da equipe reflete na quantidade de erros que serão detectados no componente escolhido da *release* candidata e reportados à equipe de desenvolvimento por turno.

Carta de Evento: Descreve uma situação (boa ou ruim) que pode ocorrer em um projeto de software. Cada carta de evento irá conter a descrição do evento e causará algum efeito ao ambiente do projeto do jogador que sorteou o evento (evento de tipo local) ou de todos os jogadores (evento de tipo global).

Carta de Conceito: Descreve um elemento que pode ser agregado ao projeto. Uma carta de conceito pode ser uma ferramenta, uma técnica ou um processo. Cada carta de conceito irá conter sua descrição e custo de implementação que deverá ser pago uma única vez.

Eclipse **PRO01** (a)


Eclipse é uma plataforma aberta para a criação de ambientes integrados de desenvolvimento (IDEs). Ela possibilita desenvolver diversos programas, aplicativos e ferramentas, de forma otimizada e padronizada, baseando-se nas iniciativas de software livre.

[Eclipse, 2009]

Qualidade: 9
verba Inicial: 400 c

Releases	CP1	CP2	CP3	CP4	Pagamento
	5F	4F	4F	2F	300 c
	3F	-	8F	6F	400 c
	5F	7F	5F	4F	500 c
	8F	5F	6F	7F	500 c
	8F	5F	10F	8F	700 c

(b) **Equipe Mega** **EQ04**



Equipe bastante dedicada ao seu trabalho e de bom conhecimento.

Salário: 70c
Adicional por Contratação: 15c
Habilidade em Desenvolvimento: 3
Tendência a Bugs: 1/5
Habilidade em Testes: 2

(c) **Subversion** **C01**


Ferramenta de controle de versão open source. Use esta carta para aumentar a produtividade das suas equipes de desenvolvimento em 2 pontos por turno. Bônus: +1 ponto de produtividade por turno, caso possua a carta C08.

[Collabnet, 2000]

Custo de Implementação: 50c

(d) **EVL01**

Dificuldades ao efetuar **merge**: caso mais de uma das suas equipes esteja trabalhando em um mesmo componente, estas equipes perdem 2 pontos de produtividade neste turno.



(e) **EVG01**

Uma multinacional entrou no mercado e oferece ótimas oportunidades de emprego. Todos os jogadores devem ceder uma de suas equipes ou aumentar o salário de todas em 1 ponto de adicional por contratação.




Figura 2. Cartas do jogo: carta de projeto (a), carta de equipe (b), carta de conceito (c), carta de evento do tipo local (d) e carta de evento do tipo global (e)

Marcadores de Funcionalidades: são utilizados para marcar no tabuleiro do jogador as funcionalidades que as equipes de desenvolvimento produzem para cada componente.

Marcadores de *Bugs*: Equipes de desenvolvimento podem produzir modificações nos componentes com *bugs*. Marcadores de *bugs* podem ser de 2 tipos:

- *Bugs* Brancos ou Defeitos: defeitos identificados pela equipe de desenvolvimento durante a produção dos componentes. Estes defeitos podem ser consertados pela equipe de desenvolvimento com o gasto de 1 ponto de habilidade gerada no turno.
- *Bugs* Cinzas ou Erros: *bugs* não identificados pela equipe de desenvolvimento durante a produção dos componentes. Estes erros são detectados pela equipe de testes na *release* candidata e são reportados à equipe de desenvolvimento com o gasto de 1 ponto de habilidade gerada no turno. Ao serem reportados à equipe de desenvolvimento, os *bugs* cinzas (erros) se transformam em *bugs* brancos (defeitos) a serem consertados.

Marcadores de Humor de Equipes: utilizados para sinalizar os turnos que uma equipe está sem receber.

Marcadores de Quantidade “I” e “V”: são utilizados para sinalizar quantos adicionais de contratação uma equipe carrega no valor total do seu salário.

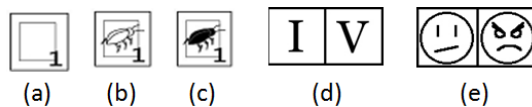


Figura 3. Marcadores do jogo: marcador de funcionalidade (a), marcador de *bug* branco (b), marcador de *bug* cinza (c), marcadores de quantidade (d) e marcadores de humor de equipes (e)

Desenvolvimento do Jogo

Início: Os jogadores rolam o dado para decidir quem começa. Quem tirar o maior número no dado é o primeiro a jogar e o fluxo do jogo segue em sentido horário. Cada jogador deve sortear uma carta de projeto que será o seu objetivo de jogo e receber a verba inicial do projeto.

Turnos: Encerrados os procedimentos iniciais, o primeiro jogador inicia o jogo que deve seguir os seguintes turnos para cada jogador:

- Turno de Contratação: o jogador pode contratar (somente) 1 equipe por turno. Ele pode contratar uma das equipes disponíveis no mercado a sua escolha pagando o salário indicado ou tentar contratar uma equipe do projeto de outro jogador aumentando o salário atual da equipe de acordo com o adicional por contratação da mesma. Para tentar contratar uma equipe de outro projeto o jogador deve efetuar uma rolagem de dado, caso tire 1, 2 ou 3 a proposta é recusada, caso tire 4, 5 ou 6 a equipe é contratada e seu valor de salário é atualizado. A equipe contratada deve

ser disposta no tabuleiro do projeto na área em que a equipe será alocada (desenvolvimento ou testes). Quaisquer realocações de função de equipes também devem ser realizadas neste turno, movendo as equipes para as áreas do tabuleiro referentes à função desejada.

- Turno de Compra: o jogador deve retirar uma carta de evento no topo da pilha. Ele deve ler em voz alta a carta de evento e realizar as ações indicadas pela carta. Caso a carta de evento se refira a todos os jogadores (evento do tipo global), todos os jogadores devem realizar as ações indicadas. Após seu uso, a carta de evento deve ser descartada.
- Turno de Produção: o jogador deve realizar as rolagens de dado referentes a cada equipe alocada em desenvolvimento que possua. De acordo com o somatório dos números tirados nas rolagens de dado para cada equipe ele pode implementar as modificações nos componentes a fim de completá-los para concluir a *release* em desenvolvimento. As modificações realizadas devem respeitar o número de tendência a *bugs* de cada equipe, logo as cartas de *bugs* devem ser sorteadas no topo da pilha respeitando o valor indicado por este atributo. Os defeitos reportados pela equipe de teste devem ser consertados neste turno pelas equipes de desenvolvimento com o gasto de 1 ponto de habilidade por defeito reportado.
- Turno de Conclusão da *Release* Candidata: caso tenha completado o número de modificações necessárias nos componentes da *release*, o jogador pode anunciar a *release* candidata e entregar seus componentes para a equipe de Teste.
- Turno de Testes: o jogador deve realizar as rolagens de dado referentes a cada equipe alocada em testes. O número tirado na rolagem de dado da equipe reflete na quantidade de *bugs* cinza (defeitos) que serão detectados nos componentes da *release* candidata e reportados a equipe de desenvolvimento.
- Turno de Conclusão da *Release*: caso a *release* candidata não possua mais *bugs* ou a sua qualidade esteja dentro do limite aceitável pelo projeto (e se o jogador assim desejar), o jogador deve anunciar a conclusão da *release* e realizar sua entrega. O jogador recebe o pagamento referente a *release*. O jogador também pode tentar entregar a *release* para o cliente mesmo se ela estiver fora dos padrões, porém, neste caso, haverá uma probabilidade de rejeição do cliente. Para a tentativa de entrega o jogador deverá efetuar uma rolagem de dados, caso o número tirado no dado seja proporcional ao número de defeitos além do permitido na qualidade do projeto a *release* é aceita. Caso contrário a *release* é rejeitada pelo cliente e o jogador não poderá tentar entregar esta *release* novamente durante 3 turnos, a não ser que esteja dentro dos padrões.
- Turno de Pagamento: o jogador deverá pagar o salário das equipes que possui. Caso o pagamento não seja efetuado o jogador sofre as seguintes penalidades:
 - 1 turno sem receber: a equipe perde 50% da sua produtividade no próximo turno de produção ou de testes.
 - 2 turnos seguidos sem receber: a equipe perde 75% da sua produtividade no próximo turno de produção ou de testes.
 - 3 turnos seguidos sem receber: o jogador perde a equipe e ela retorna à área de mercado. O jogador fica impedido de recontratar esta equipe.

Marcadores de humor de equipe devem ser utilizados para demarcar os turnos sem pagamento de uma equipe.

Conclusão: Acontece a última rodada do jogo quando o primeiro jogador concluir a última *release* do seu projeto. Caso nenhum outro jogador conclua a sua última *release* nesta rodada, o primeiro jogador que concluiu é declarado vencedor do jogo. Caso outro jogador também conclua a sua última *release* nessa rodada, o vencedor do jogo é aquele que possuir mais dinheiro.

LISTAS DE REFERÊNCIAS

Referências Utilizadas:

Ant, Apache Software Foundation, 1999. Disponível em: <http://www.apache.org/>. Acesso em: 18 Nov. 2009.
Carta(s): C03

Berczuk, S. *Pragmatic Software Configuration Management*. IEEE Softw., v.20, n.2, p.15-17. 2003.
Carta(s): C05, C06, C010

Boehm, B.W., Basili, V.R. *Software Defect Reduction Top 10 List*. IEEE Computer 34(1), Los Alamitos, CA: IEEE Computer Society, 2001, pages 135-137.
Carta(s): C08

Clam Antivirus. Disponível em: <http://www.clamav.net/>. Acesso em: 18 Nov. 2009.
Carta(s): PRO02

Eclipse. Disponível em: <http://www.eclipse.org/>. Acesso em: 18 Nov. 2009.
Carta(s): PRO01

Estublier, J. *Objects Control for Software Configuration Management*. International Conference on Advanced Information Systems Engineering (CAiSE), Interlaken, Switzerland, p. 359-373, 2001.
Carta(s): C04

IEEE Std 828 - *IEEE Standard for Software Configuration Management Plans*, Institute of Electrical and Electronics Engineers, 2005.
Carta(s): C07

ISO/IEC 17799 - *Information technology, Code of practice for information security management*, 2000.
Carta(s): C09

Mozilla Community Website. Disponível em: <http://www.mozilla.org/>. Acesso em: 18 Nov. 2009.
Carta(s): PRO03

OpenOffice.org. Disponível em: <http://pt.openoffice.org/>. Acesso em: 18 Nov. 2009.
Carta(s): PRO04

Snort. Disponível em: <http://www.snort.org/>. Acesso em: 18 Nov. 2009.
Carta(s): PRO05

Sorio, W. *O que é benchmarking?* Disponível em: <http://www.guiarh.com.br/z59.htm>. Acesso em: 18 Nov. 2009.
Carta(s): EVG06

Subversion, Collabnet, 2000. Disponível em: <http://www.collab.net/>. Acesso em 18 de Nov. de 2009.

Carta(s): C01

Trac, Edgewall Software, 2003. Disponível em: <http://trac.edgewall.org/>. Acesso em: 18 Nov. 2009.

Carta(s): C02

Leituras Recomendadas:

Dart, S. Concepts in Configuration Management Systems. International Workshop on Software Configuration Management (SCM). Trondheim, Norway: ACM Press. June, 1991. 1-18 p.

Estublier, J. Software Configuration Management: a Roadmap. International Conference on Software Engineering (ICSE), The Future of Software Engineering. Limerick, Ireland. June, 2000. 279-289 p.

Mens, T. A State-of-the-Art Survey on Software Merging. IEEE Trans. Softw. Eng., v.28, n.5, p.449-462. 2002.

Walrad, C. and Strom, D. The Importance of Branching Models in SCM. IEEE Computer, v.35, n.9, September, p.31-38 2002.

GLOSSÁRIO

Análise de Impacto: A análise visa relatar os impactos em custo, cronograma, funcionalidades, etc. da implementação de uma solicitação de modificação.

Backup: Cópia de segurança de dados para que estes possam ser restaurados caso ocorra algum imprevisto.

Benchmarking: Processo contínuo de comparação de produtos / serviços / práticas empresarias entre os mais fortes concorrentes ou empresas. É um processo de pesquisa que permite realizar comparações "companhia-a-companhia" para identificar o melhor do melhor e alcançar um melhor nível de superioridade. (Sorrio, 2009)

Check-in: Método utilizado para enviar os itens de configuração que estão na área de trabalho do desenvolvedor para o repositório.

Check-out: Método utilizado para trabalhar com itens de configuração que estão no repositório. Este método copia as informações do repositório para a área de trabalho do desenvolvedor.

Classificação das Solicitações: A classificação visa priorizar modificações mais importantes. O critério de classificação de uma solicitação de modificação deve estar explicitado no plano de gerência de configuração. Exemplos de classificações: críticas, fatais, não fatais, cosméticas, etc.

Commits: Modificações feitas pelo usuário e enviadas ao repositório.

Configuração: O estado em que um sistema ou determinado item se encontra em um determinado momento.

Documentação: A documentação de software descreve cada parte do código fonte, geralmente uma função, uma classe, um simples trecho ou módulo. Consiste também no conjunto de manuais gerais e técnicos, além de diagramas explicando o funcionamento de um software como um todo ou cada parte dele.

Evolução de um Produto: As alterações no estado de um produto através do tempo.

Ferramenta de Controle de Modificações: Software que tem por objetivo ajudar o usuário a rastrear modificações, entender o porquê de cada modificação e qual o seu impacto no projeto como um todo.

Ferramenta de Controle de Versão (ou versionamento): É um software com a finalidade de gerenciar diferentes versões no desenvolvimento de um documento qualquer. Essas ferramentas são comumente utilizadas no desenvolvimento de software para controlar as diferentes versões – histórico e desenvolvimento – dos códigos-fontes e também da documentação de sistemas.

Gerência de Configuração de Software: “É uma disciplina para o controle da evolução de sistemas de software.” Susan Dart, 1991.

Item de Configuração: É o elemento básico da gerência de configuração. Pode ser um documento, código, plano, *hardware*. O conjunto de itens de configuração determina a configuração de um sistema.

Merge: A mesclagem (do inglês, *merge*) consiste na fusão automática de versões através da comparação das diferenças entre elas.

Modelo ou Estratégia de Ramificação: Em uma ferramenta de controle de versões é possível quebrar a linha do desenvolvimento principal em mais de um caminho. Este processo, chamado de ramificação, é muito útil quando se conquista uma versão estável (*baseline*) ou quando se deseja realizar uma tentativa "não convencional" fora do ramo principal. Existem diferentes estratégias de criação de ramos que podem ser utilizadas e combinadas.

Políticas de Controle de Concorrência: Conjunto de regras estabelecidas para gerenciar a utilização de itens de configuração simultaneamente e assim evitar problemas de sincronização que podem causar perda de consistência, erro nos dados e perda de atualizações.

Release: (1) Uma versão particular de um item de configuração feito para um propósito específico (ex.: *release* de teste). (2) Uma notificação e distribuição formal de uma versão aprovada do sistema.

Repositório: Local onde são armazenadas versões e *releases* de um sistema e seus itens de configuração.

Solicitação Emergencial: Uma solicitação de modificação de alta urgência, i.e. de prioridade absoluta.