UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

ESCOLA DE INFORMÁTICA APLICADA

JOGO PARA ENSINO DE PROCESSO DE SOFTWARE

VICTOR MOURA

Orientador

Gleison Santos

RIO DE JANEIRO, RJ – BRASIL

DEZEMBRO DE 2014

JOGO PARA ENSINO DE PROCESSO DE SOFTWARE

VICTOR MOURA

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovada por:

_____

GLEISON SANTOS, D.Sc. (UNIRIO)

_____

MARIANO PIMENTEL, D.Sc. (UNIRIO)

_____

CLÁUDIA CAPELLI, D.Sc. (UNIRIO)

RIO DE JANEIRO, RJ – BRASIL.

DEZEMBRO DE 2014

Agradecimentos

Agradeço a todas as pessoas que puderam me acompanhar até o fim dessa jornada, mas principalmente a àqueles que não puderam mas sempre me desejaram o melhor na vida que são meus avós, e minha mãe.

Agradeço também a toda equipe da UNIRIO, entre servidores e professores, os quais sem eles não seria possível chegar ao fim e com o conhecimento necessário.

Outros agradecimentos vão para todos os amigos, parentes colegas que me ajudaram ao longo da faculdade, sejam em termos de matérias ou psicológicos.

Por fim, agradeço ao meu orientador Gleison Santos por toda a paciência, e apoio na conclusão deste trabalho e a todos os amigos que eu pude fazer ao longo desses anos nesta universidade.

RESUMO

Esta monografia tem como objetivo a criação de um jogo educativo para o ensino de Processos de Software chamado ProcSoft. Este jogo baseia se em normas como a ISO/IEC 29110, para a criação de processos para pequenas entidades, e também em conceitos gerais da Engenharia de Software. O objetivo do jogo é promover a melhor assimilação de conceitos sobre Processos de Software de uma forma lúdica. Esta monografia também apresenta os resultados da avaliação da primeira versão do jogo experimentado em uma turma de graduação, onde se evidenciou os resultados positivos dessa experiência em relação ao aprendizado através de jogos educacionais.

Palavras-chave: Processos de Software, Engenharia de Software, Jogo para Processos de Software.

ABSTRACT

In this work we present an educational game to teach Software Process named ProcSoft. ProcSoft is based on quality standards such as ISO/IEC 29110, that aims to support the definition of software processes to very small settings and also on general Software Engineering concepts. ProcSoft goal is to promote the assimilation of software process concepts. We also present the results of ProcSoft first version evaluation executed in a undergraduation class. There were positive results regarding the students' learning.

Keywords: Software Processes, Software Engineering, Software Process Games.

Índice

Índice de Tabelas

Índice de Figuras

1. Introdução

1.1 Contexto

Ao utilizar jogos como ferramenta de aprendizado é mostrado que existem outros métodos de aprendizado, além da estrutura clássica de exposição da matéria através de apresentações de teoria, *slides*, testes e provas. Jogos sérios ou educativos, como são chamados, têm como objetivo instruir e educar os jogadores sobre algum tema ou conceito. Além disso, estimulam os alunos na construção de seu próprio conhecimento por meio de suas interações [Tarouco *et al.,* 2004] e tornam o aprendizado menos cansativo e mais eficiente na retenção do conhecimento, estimulando, assim, o aprendizado.

As pessoas possuem familiaridade com jogos desde crianças, e, ao participarem de um jogo, são remetidas a uma sensação de diversão, o que estimula a cognição e amplia a capacidade de aprendizado. Os jogos, de maneira geral, compreendem um conjunto de atividades que podem envolver um ou mais participantes e comumente desafiam as habilidades físicas e/ou mentais dos jogadores, exigindo que esses sigam um conjunto de regras para alcançar um determinado objetivo previamente estabelecido [Dempsey *et al.*, 1993].

1.2 Motivação

Os jogos educacionais são projetados para ensinar determinado assunto, expandir conceitos, reforçar o desenvolvimento, compreender um acontecimento histórico ou cultural, ou auxiliar na aprendizagem de uma habilidade [Yee, 2006]. Os jogos educacionais podem reduzir o tempo de formação, oferecer mais oportunidades para a prática e, consequentemente, aumentar a aquisição de conhecimentos [Brownfield e Vick, 1983; Ricci, 1994]. Os jogos também têm-se mostrado eficazes quando desenvolvidos para ensinar uma determinada habilidade [Griffiths, 2002] e para

estimular a motivação e o interesse do aluno [Malone, 1980; Malone, 1983; Dempsey *et al*., 1994].

O uso de jogos educativos para o ensino de Engenharia de Software vem ganhando espaço. Na literatura, por exemplo, no Jogo das Sete Falhas [Diniz e Dazzi, 2011] e no jogo a Ilha dos Requisitos [Gonçalves *et al.*, 2010], é possível encontrar relatos de jogos sobre engenharia de software e, em especial, também sobre processos de software. Segundo Pressman [2011], processo de software é definido como "uma metodologia para as atividades, ações e tarefas necessárias para desenvolver um software de alta qualidade".

Jogos como JEES [Figueiredo *et al.*, 2010] e Ilha dos Requisitos [Gonçalves *et al.*, 2010] aplicam os conceitos diretamente relacionados às práticas dos principais modelos de maturidade para melhoria do processo de software, como o MR-MPS-SW [Softex, 2009] e o CMMI [SEI, 2006], ou associadas a normas internacionais como a ISO/IEC 12207 [ISO/IEC, 2008] (que apresenta os processos do ciclo de vida do software) ou a ISO/IEC 29110 [ABNT, 2011] (que tem por objetivo introduzir conceitos de processos, ciclo de vida e normalização e é indicada para pequenas empresas).

No entanto, não foram encontrados relatos de jogos específicos para o ensino da estrutura padrão de dos processos de software e das principais atividades que deveriam compô-los.

1.3 Objetivos

Dessa forma, o foco abordado neste trabalho é a criação de um jogo de Engenharia de Software que tem como ideia central a criação de um Processo de Software, usando como referência a norma internacional ISO/IEC 29110 [ABNT, 2011] como base.

Por meio desse jogo é esperado que conceitos por trás dos Processos de Software sejam mais bem assimilados pelos estudantes, promovendo maior absorção do conteúdo.

1.4 Organização do texto

A estrutura dessa monografia está dividida em quatro capítulos, além desta introdução:

- no Capítulo 2 é descrito genericamente o que é Engenharia de Software, como um Processo de Software funciona e são apresentadas normas para definição de processos e modelos de maturidade de desenvolvimento de software.
- no Capítulo 3 serão discutidos jogos educacionais e, em especial, jogos relacionados à Engenharia de Software.
- na Seção 4 será apresentada a proposta do jogo e como ele foi aplicado em sala de aula.
- por fim, a Seção 5 apresenta as conclusões e os trabalhos futuros.

2. Engenharia e Processos de Software

2.1 Introdução

Para a boa compreensão do jogo que será proposto nesta monografia, é necessário entender alguns conceitos sobre engenharia de software. Dessa forma, este capítulo apresentará o que é Engenharia de Software e Processos de Software, além de apresentar modelos de maturidade de software e normas para a criação e definição de um processo de software.

2.2 Engenharia de Software

Existem diversas definições sobre o que é Engenharia de Software, que foram surgindo ao longo do tempo. Como definição inicial temos a proposta por Fritz Bauer [Naur, 1969]:

- Engenharia de Software é o estabelecimento e o emprego de sólidos princípios de engenharia de modo a obter software de maneira econômica, que seja confiável e funcione de forma eficiente em máquinas reais.

Uma definição posterior foi definida pela IEEE [1993], que seria uma definição mais abrangente:

- Engenharia de Software é a aplicação de uma abordagem sistemática, disciplinada e quantificável no desenvolvimento, na operação e na manutenção do software, isto é, a aplicação de engenharia ao software.

Para Pressman [2011], a engenharia de software é uma tecnologia em camadas, como pode ser visto na Figura 1.

Figura 1 - Camadas de Engenharia de Software [Pressman, 2011]

A base da engenharia de software é a camada de *processo*. O processo é o que mantém as camadas coesas e possibilita o desenvolvimento do software de forma racional e dentro do prazo. O processo define uma metodologia que deve ser estabelecida para a entrega efetiva de tecnologia de engenharia de software. O processo de software constitui a base para o controle do gerenciamento de projetos e estabelece o contexto no qual são aplicados métodos técnicos, são gerados produtos derivados (modelos, documentos, dados, etc.), são estabelecidos marcos, a qualidade é garantida e mudanças são geridas de forma apropriada.

Os *métodos* da engenharia de software fornecem as informações técnicas para desenvolver software. Os métodos envolvem uma ampla gama de tarefas, que incluem: comunicação, análise de requisitos, modelagem de projeto, construção de programa, testes e suporte. Estes métodos baseiam-se em um conjunto de princípios básicos que governam cada área da tecnologia e incluem atividades de modelagem e outras técnicas descritivas.

As *ferramentas* da engenharia de software fornecem apoio automatizado ou semiautomatizado para o processo e para os métodos. Quando as ferramentas são integradas de modo que as informações criadas por uma ferramenta possam ser usadas por outra, é estabelecido um sistemas para suporte ao desenvolvimento de software.

2.3 Processos de Software

Segundo Pressman [2011], processo é um conjunto de atividades, ações e tarefas realizadas na criação de um produto de trabalho. Uma atividade esforça-se para atingir um objetivo amplo e é utilizada independente do campo de aplicação, do tamanho do projeto, da complexidade ou do grau de rigor que a engenharia de software será aplicada. Uma ação envolve um conjunto de tarefas que resultam em um artefato de

software fundamental. Uma tarefa se concentra em um objetivo pequeno, porém bem definido e produz um resultado tangível.

Dessa forma, o Processo de Software é uma abordagem adaptável que possibilita à equipe de software realizar o trabalho de selecionar e escolher o conjunto apropriado de ações e tarefas com o objetivo de sempre entregar o software dentro do prazo e com qualidade suficiente para satisfazer àqueles que patrocinaram a sua criação e àqueles que irão utilizá-lo [Pressman, 2011].

A metodologia de um processo estabelece as condições necessárias para um processo de engenharia de software completo, por meio da identificação de um pequeno número de atividades estruturais aplicáveis a todos os projetos de software, independente de tamanho ou complexidade. Além disso, a metodologia de processo engloba um conjunto de atividades de apoio aplicáveis e todo o processo de software [Pressman, 2011], como pode ser visto na Figura 2.

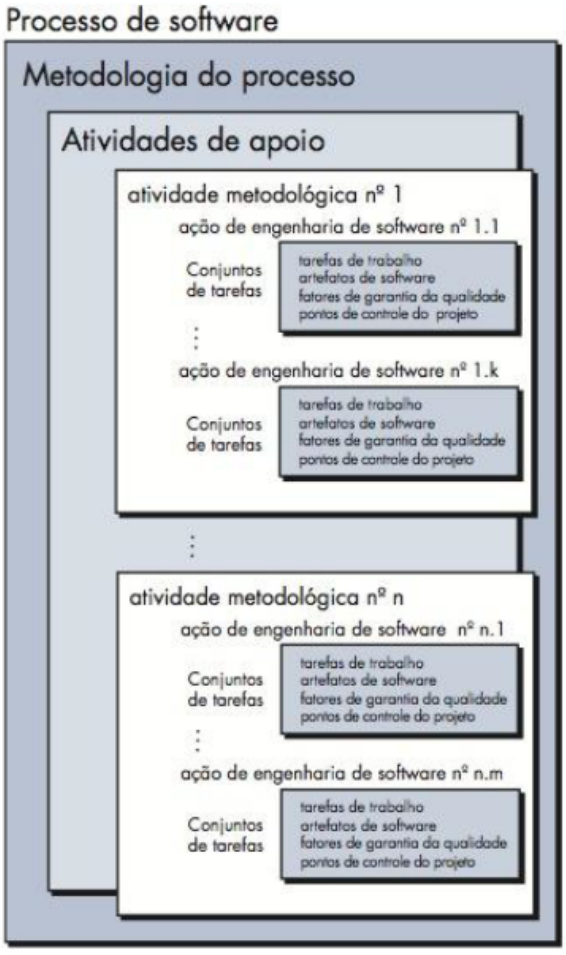


Figura 2 - Uma metodologia do Processo de Software [Pressman, 2011]

Uma metodologia de processo genérica para engenharia de software compreende cinco atividades [Pressman, 2011]: Comunicação, Planejamento, Modelagem, Construção e Entrega. Essas atividades podem ser utilizadas para a construção de programas pequenos e simples, para a criação de grandes aplicações para a internet e para a engenharia de grandes e complexos sistemas de informação. Os detalhes serão bem diferentes nos casos, mas as atividades metodológicas permaneceram as mesmas.

As atividades metodológicas do processo de software são complementadas por uma série de atividades de apoio, que em geral são aplicadas ao longo de um projeto, ajudando a equipe a gerenciar, a controlar o progresso, a qualidade, as mudanças e os riscos. As atividades de apoio costumam ser [Pressman, 2011]: Controle e acompanhamento do projeto, Administração de riscos, Garantia da qualidade de software, Revisões técnicas, Medição, Gerência da configuração de software, Gerência de reúso, Preparo e produção artefato de software.

Pode-se dizer que o processo de software não é rígido e nem deve ser. Ele deve ser ágil e adaptável (ao problema, ao projeto, à equipe). Ou seja, um processo adotado para um determinado projeto pode ser muito diferente de um adotado para outro.

2.4 Normas para Definir Processos de Software

A qualidade de um produto de *software* está fortemente relacionada com a qualidade de seu processo de produção seguido por quem o desenvolve. Quando ainda não se tem um produto, o seu processo de desenvolvimento deverá conferir a capacidade de satisfazer as necessidades do cliente final. Entendido que a qualidade do *software* produzido é ponto preponderante para a competitividade das empresas e que para assegurar-se tal qualidade deve-se garantir a qualidade de seu processo de desenvolvimento, passou-se então a criar e estabelecer normas, padrões, técnicas organizacionais e modelos de referência para implementação de bons processos de desenvolvimento de *software* [Fuggetta, 2000].

Nesse contexto, as normas internacionais na área de engenharia de *software* indicam as boas práticas, métodos reconhecidamente eficazes e processos sólidos, testados e confiáveis.

2.4.1 Norma ISO/IEC 12207: Tecnologia de informação – Processos de Ciclo de Vida de Software

A norma ISO/IEC 12207 [ISO/IEC, 2008] contém um conjunto de processos, atividades e tarefas projetado para ser adaptado de acordo com cada projeto de *softwar*e. A estrutura cobre o ciclo de vida do software desde a concepção de ideias até a descontinuação do software.

A norma reconhece que nem todos os projetos e organizações precisam usar todos os processos providos. Assim, a implementação do padrão tipicamente envolve a seleção de processos apropriados para o projeto/organização [ISO/IEC, 2008]. O processo de adaptação consiste na supressão de processos, atividades e tarefas não aplicáveis. As partes envolvidas são responsáveis pela seleção de um modelo de ciclo de vida para o projeto e pelo mapeamento dos processos, atividades e tarefas dentro desse modelo. As partes envolvidas são também responsáveis pela seleção e aplicação dos métodos e pela execução das atividades e tarefas adequadas ao projeto (ISO/IEC, 2008).

Na norma, processos são descritos por meio de nome, propósito e resultados esperados. Um processo também pode ser decomposto em atividades ou processos de nível mais baixo (visando deixar mais claro seus objetivos). Uma atividade é uma coleção de tarefas. Uma tarefa é uma ação que tem por intenção apoiar a obtenção de resultados. Pode ser um requisito, uma recomendação ou uma permissão. Os propósitos e os resultados dos processos constituem um Modelo de Referência de Processo, que podem ser utilizados para comporem, por exemplo, modelos de avaliação ou de maturidade (como será visto na próxima seção).

Os processos da ISO/IEC 12207 são agrupados em sete categorias, como pode ser visto a seguir. Além da categorização anterior, há duas subdivisões maiores: **Processos de Contexto de Sistema (**que tratam de um produto ou serviço de software ou um sistema de software que opera independentemente) e **Processos Específicos de Software** (para uso na implementação de um produto ou serviço de software que é um elemento de um sistema maior).

Processos de Acordo - Definem as atividades necessárias para se estabelecer um acordo entre duas organizações. A Tabela 1 apresenta os Processo de Acordo.

Tabela 1 – Processos de Acordo

Processo	Propósito
Processo de Aquisição	Obter um produto ou serviço que satisfaça a necessidade expressa pelo cliente.
Processo de Fornecimento	Fornecer um produto ou serviço ao cliente que atenda aos requisitos acordados.

Processos Organizacionais de Habilitação de Projetos – Gerenciam a capacidade organizacional de adquirir e fornecer produtos ou serviços por meio de projetos, como podem ser vistos na Tabela 2.

Tabela 2 – Processos Organizacionais de Habilitação de Projetos

Processo	Propósito
Processo de Gerência de Modelo de Ciclo de Vida	Definir, manter e assegurar a disponibilidade de políticas, processos, modelos de ciclo de vida e procedimentos para uso da organização.
Processo de Gerência de Infraestrutura	Prover a infraestrutura e os serviços necessários para os projetos.
Processo de Gerência de Portfólios de Projetos	Iniciar e sustentar os projetos necessários, suficientes e adequados, de modo a atender aos objetivos estratégicos da organização.
Processo de Gerência de Recursos Humanos	Prover à organização os recursos humanos necessários e manter suas competências consistentes com as necessidades de negócio.
Processo de Gerência de Qualidade	Fornecer garantia de que os produtos, serviços e implementações de processos de ciclo de vida satisfazem objetivos de qualidade organizacionais e atingem a satisfação do cliente.

Processos de Projeto – Há duas subcategorias de processos de projeto: **Processos de Gerência de Projetos,** usados para planejar, executar, avaliar e controlar o progresso de um projeto (Tabela 3); e **Processos de Apoio de Projeto**, que apoiam objetivos de gerência especializados (Tabela 4).

Tabela 3 – Processos de Gerência de Projetos

Processo	Propósito
Processo de Planejamento de Projeto	Produzir e comunicar planos de projeto efetivos e passíveis de serem trabalhados.
Processo de Controle e Avaliação de Projeto	Determinar o status do projeto e garantir que o projeto está sendo realizado de acordo com os planos e cronogramas, dentro do orçamento e que ele satisfaz objetivos técnicos, o que pode envolver replanejamento quando apropriado.

Tabela 4 – Processos de Apoio de Projeto

Processo	Propósito
Processo de Gerência de Decisão	Selecionar o curso mais benéfico da ação do projeto, quando existirem alternativas.
Processo de Gerência de Riscos	Identificar, analisar, tratar e monitorar riscos continuamente.
Processo de Gerência de Configuração	Estabelecer e manter a integridade de todos os produtos de trabalho de um processo ou projeto e disponibilizá-los às partes envolvidas.
Processo de Gerência de Informação	Prover informação relevante, oportuna, completa, válida e, se requerido, confidencial às partes designadas.
Processo de Medição	Coletar, analisar e reportar dados relativos a produtos desenvolvidos e processos implementados, para apoiar a gerência efetiva dos processos e para demonstrar objetivamente a qualidade dos produtos.

Processos Técnicos - Usados para (i) definir os requisitos do sistema; (ii)transformá-los em um produto efetivo; (iii) permitir a reprodução consistente do produto onde necessário; (iv) usar o produto; (v) prover os serviços requeridos e sustentá-los; (vi) descartar o produto quando ele é retirado de serviço. A Tabela 5 apresenta os Processos Técnicos.

Tabela 5 – Processos Técnicos

Processo	Propósito
Processo de Definição dos Requisitos dos *Stakeholders*	Definir os requisitos para um sistema que pode prover serviços requeridos por usuários ou outros *stakeholders*.
Processo de Análise de Requisitos de Sistema	Transformar os requisitos de *stakeholder* definidos em um conjunto de requisitos técnicos de um sistema.
Processo de Projeto Arquitetural do Sistema	Identificar quais requisitos do sistema devemser alocados a quais elementos do sistema.
Processo de Implementação	Conceber os elementos de sistema especificados.
Processo de Integração do Sistema	Integrar os elementos de sistema, incluindo itens de software, hardware, operações manuais e outros sistemas, se necessário, para produzir um sistema completo.
Processo de Teste de Qualificação de Sistema	Garantir que a implementação de cada requisito de sistemas foi testada e que o sistema está pronto para entrega.
Processo de Instalação de Software	Instalar o produto de software no ambiente alvo.
Processo de Apoia à Aceitação do Sistema	Assistir o comprador a atingir a confiança de que o produto satisfaz os requisitos.
Processo de Operação do Software	Operar o produto de software em seu ambiente e prover suporte aos clientes.
Processo de Manutenção de Software	Prover apoio efetivo em termos de custo a um produto de software entregue.
Processo de Descarte de Software	Concluir a existência de uma entidade de software de um sistema.

Processos de Implementação de Software – Os dois processos deste grupo são usados para produzir um elemento de sistema específico implementado em software. O Processo de Implementação de Software tem como propósito é produzir um elemento

de sistema especificado, implementado como um produto ou serviço de software. O Processo de Implementação de Software é um processo geral, decomposto em outros processos de nível mais baixo, como pode ser visto na Tabela 6.

Tabela 6 – Processos de Implementação de Software

Processos	Propósitos
Processo de Análise de Requisitos de Software	Estabelecer os requisitos dos elementos de software do sistema
Processo de Projeto Arquitetural de Software	Prover um projeto (design) para o software que implemente os requisitos e que possa ser verificado junto a eles
Processo de Projeto Detalhado de Software	Prover um projeto (design) para o software que implemente os requisitos e a arquitetura, e que possa ser verificado junto a eles, em um nível de detalhes suficiente para que possa ser codificado e testado
Processo de Construção de Software	Produzir unidades de software executável que adequadamente reflitam o projeto (design) de software
Processo de Integração de Software	Combinar as unidades e componentes de software, produzindo itens de software integrados, consistentes com o projeto (design) de software e que demonstrem que requisitos funcionais e não funcionais foram satisfeitos em uma plataforma operacional completa ou equivalente
Processo de Teste de Qualificação de Software	Confirmar que o produto de software integrado satisfaz os requisitos definidos

Processos de Apoio de Software - Auxiliam o Processo de Implementação de Software, contribuindo para o sucesso e a qualidade do projeto de software (como pode ser visto na Tabela 7).

Tabela 7 – Processos de Apoio de Software

Processo	Propósito
Processo de Gerência de Documentação de Software	Desenvolver e manter as informações de software produzidas por um processo.
Processo de Gerência de Configuração de Software	Estabelecer e manter a integridade dos itens de software de um processo ou projeto e disponibilizá-los às partes interessadas.
Processo de Garantia da Qualidade de Software	Prover garantia de que produtos de trabalho e processos estão de acordo com planos e cláusulas pré-definidas.
Processo de Verificação de Software	Confirmar que cada produto de trabalho ou serviços de software de um processo ou projeto reflete adequadamente os requisitos especificados.
Processo de Validação de Software	Confirmar se os requisitos de um produto de trabalho para um uso específico pretendido são satisfeitos.
Processo de Revisão de Software	Manter um entendimento comum com os *stakeholders* acerca do processo em relação aos objetivos.
Processo de Auditoria de Software	Determinar de forma independente a conformidade de produtos e processos selecionados em relação a requisitos, planos e acordos.
Processo de Resolução de Problemas de Software	Assegurar que todos os problemas encontrados são identificados, analisados, gerenciados e controlados até a resolução.

Processos de Reutilização de Software - Visam trabalhar a capacidade de uma organização reutilizar itens de software além das fronteiras dos projetos. Pela sua

natureza, operam fora das fronteiras de qualquer projeto particular. A Tabela 8 apresenta os processos de reutilização de software.

Tabela 8 – Processos de Reutilização de Software

Processo	Propósito
Processo de Engenharia de Domínio	Desenvolver e manter modelos, arquiteturas e ativos de domínio.
Processo de Gerência de Ativos e Reuso	Gerenciar a vida dos ativos reutilizáveis desde sua concepção até sua retirada.
Processo de Gerência de Programa de Reuso	Planejar, estabelecer, gerenciar, controlar e monitorar um programa de reuso da organização sistematicamente explorar oportunidades de reuso.

Além dos processos de ciclo de vida, a ISO/IEC 12207, em seu anexo A, define um **processo de adaptação**, conforme dito anteriormente. O propósito deste processo é adaptar os processos da norma para satisfazer circunstâncias particulares. As atividades do Processo de Adaptação incluem: (i) identificação do ambiente do projeto; (ii) solicitação de Informações; (iii) seleção de processos, atividades e tarefas; (iv) documentação de decisões e motivos da adaptação.

2.3.2 MPS.BR

O MPS.BR [Softex, 2009] é um programa mobilizador que foi criado em 2003 pela Softex para melhorar a capacidade de desenvolvimento de software nas empresas brasileiras. A iniciativa foi responsável pelo desenvolvimento do Modelo de Referência para Melhoria do Processo de Software Brasileiro (MR-MPS-SW), que levou em consideração normas e modelos internacionalmente reconhecidos, boas práticas da engenharia de software e as necessidades de negócio da indústria de software nacional. Com o MR-MPS-SW foi possível estabelecer um caminho economicamente viável para que organizações, incluindo as pequenas e médias empresas, alcancem os benefícios da melhoria de processos e da utilização de boas práticas da engenharia de software em um intervalo de tempo razoável. Ele trouxe para a indústria nacional ganhos comprovados de competitividade, por isso é considerado um marco que representa a evolução da qualidade do software desenvolvido no país.

O Modelo de Referência MPS para Software (MR-MPS-SW) define níveis de maturidade que são uma combinação entre processos e sua capacidade. A definição dos processos segue os requisitos para um modelo de referência de processo apresentados na

ISO/IEC 15504-2, declarando o propósito e os resultados esperados de sua execução. Isso permite avaliar e atribuir graus de efetividade na execução dos processos. As atividades e tarefas necessárias para atender ao propósito e aos resultados esperados não são definidas neste guia, devendo ficar a cargo dos usuários do MR-MPS-SW.

A capacidade do processo é a caracterização da habilidade do processo para alcançar os objetivos de negócio, atuais e futuros; estando relacionada com o atendimento aos atributos de processo associados aos processos de cada nível de maturidade.

Os níveis de maturidade estabelecem patamares de evolução de processos, caracterizando estágios de melhoria da implementação de processos na organização. O nível de maturidade em que se encontra uma organização permite prever o seu desempenho futuro ao executar um ou mais processos. O MR-MPS-SW define sete níveis de maturidade: A (Em Otimização), B (Gerenciado Quantitativamente), C (Definido), D (Largamente Definido), E (Parcialmente Definido), F (Gerenciado) e G (Parcialmente Gerenciado). A escala de maturidade se inicia no nível G e progride até o nível A. Para cada um destes sete níveis de maturidade é atribuído um perfil de processos que indicam onde a organização deve colocar o esforço de melhoria. O progresso e o alcance de um determinado nível de maturidade do MR-MPS-SW se obtêm quando são atendidos os propósitos e todos os resultados esperados dos respectivos processos e os resultados esperados dos atributos de processo estabelecidos para aquele nível. A Tabela 9 apresenta a estrutura dos níveis de maturidade do MR-MPS-SW.

Tabela 9 - Níveis de maturidade do MR-MPS-SW

Nível	Processos	Definição
A	Este nível não possui processos específicos.	
B	Gerência de Projetos – GPR (evolução)	No nível B, a gerência de projetos passa a ter um enfoque quantitativo, refletindo a alta maturidade que se espera da organização. Novamente, alguns resultados evoluem e outros são incorporados.
C	Gerência de Riscos – GRI	Identificar, analisar, tratar, monitorar e reduzir continuamente os riscos em nível organizacional e de projeto.
	Desenvolvimento para Reutilização – DRU	Identificar oportunidades de reutilização sistemática de ativos na organização e, se possível, estabelecer um programa de reutilização para desenvolver ativos a partir de engenharia de domínios de aplicação.
	Gerência de Decisões – GDE	Analisar possíveis decisões críticas usando um processo formal, com critérios estabelecidos, para avaliação das alternativas identificadas.
D	Verificação – VER	Confirmar que cada serviço e/ou produto de trabalho do processo ou do projeto atende apropriadamente os requisitos especificados.
	Validação – VAL	Confirmar que um produto ou componente do produto atenderá a seu

Nível	Processos	Definição
		uso pretendido quando colocado no ambiente para o qual foi desenvolvido.
	Projeto e Construção do Produto – PCP	Projetar, desenvolver e implementar soluções para atender aos requisitos.
	Integração do Produto – ITP	Compor os componentes do produto, produzindo um produto integrado consistente com seu projeto, e demonstrar que os requisitos funcionais e não-funcionais são satisfeitos para o ambiente alvo ou equivalente.
	Desenvolvimento de Requisitos – DRE	Os requisitos do cliente, do produto e dos componentes do produto.
E	Gerência de Projetos – GPR (evolução)	Assim, a partir do nível E, alguns resultados evoluem e outros são incorporados, de forma que a gerência de projetos passe a ser realizada com base no processo definido para o projeto e nos planos integrados.
	Gerência de Reutilização – GRU	Gerenciar o ciclo de vida dos ativos reutilizáveis.
	Gerência de Recursos Humanos – GRH	Prover a organização e os projetos com os recursos humanos necessários e manter suas competências adequadas às necessidades do negócio.
	Definição do Processo Organizacional – DFP	Estabelecer e manter um conjunto de ativos de processo organizacional e padrões do ambiente de trabalho usáveis e aplicáveis às necessidades de negócio da organização.
	Avaliação e Melhoria do Processo Organizacional – AMP	Determinar o quanto os processos padrão da organização contribuem para alcançar os objetivos de negócio da organização e para apoiar a organização a planejar, realizar e implantar melhorias contínuas nos processos com base no entendimento de seus pontos fortes e fracos.
F	Medição – MED	Coletar, armazenar, analisar e relatar os dados relativos aos produtos desenvolvidos e aos processos implementados na organização e em seus projetos, de forma a apoiar os objetivos organizacionais.
	Garantia da Qualidade – GQA	Assegurar que os produtos de trabalho e a execução dos processos estejam em conformidade com os planos, procedimentos e padrões estabelecidos.
	Gerência de Portfólio de Projetos – GPP	Iniciar e manter projetos que sejam necessários, suficientes e sustentáveis, de forma a atender os objetivos estratégicos da organização.
	Gerência de Configuração – GCO	Estabelecer e manter a integridade de todos os produtos de trabalho de um processo ou projeto e disponibilizá-los a todos os envolvidos.
	Aquisição – AQU	Gerenciar a aquisição de produtos que satisfaçam às necessidades expressas pelo adquirente.
G	Gerência de Requisitos – GRE	Gerenciar os requisitos do produto e dos componentes do produto do projeto e identificar inconsistências entre os requisitos, os planos do projeto e os produtos de trabalho do projeto.
	Gerência de Projetos – GPR	Estabelecer e manter planos que definem as atividades, recursos e responsabilidades do projeto, bem como prover informações sobre o andamento do projeto que permitam a realização de correções quando houver desvios significativos no desempenho do projeto. O propósito deste processo evolui à medida que a organização cresce em maturidade.

2.3.3 ISO/IEC 29110

A norma internacional ISO/IEC 29110 [ABNT, 2011] é um instrumento que pode propiciar às micro e pequenas empresas (MPE) desenvolvedoras de produtos e serviços de software a ter um processo produtivo de maior qualidade e, com isso,

aumentar a satisfação dos seus clientes, sua competitividade e sua capacidade de acessar novos mercados. Apesar de foco inicial da norma ter sido as micro e pequenas empresas, a sua versão final está endereçada a pequenas entidades (ou VSE, do inglês *very small entities*), que podem ser, por exemplo, um departamento ou equipe dentro de uma grande organização ou, até mesmo, uma pequena empresa.

Dessa forma, a série ISO/IEC 29110 – Engenharia de Software – Perfis de Ciclo de Vida para Micro-Organizações – tem como público-alvo as micro-organizações e as VSE. Seu propósito maior é fazer com que essas organizações alcancem seus objetivos de qualidade, sem, necessariamente, ter que demandar projetos de longo prazo e altos investimentos para adoção das normas relevantes ao seu contexto.

Para as pequenas empresas, que em sua maioria convive com restrições financeiras relevantes, além da complexidade, outros fatores também colaboram para a não adoção das normas, entre eles: falta de recursos (financeiros e humanos), alto custo e longa duração para os projetos de implantação de melhoria de processos baseados em normas e modelos. Dessa forma, com o objetivo de dirimir estas dificuldades e propiciar ao contexto específico das micro e pequenas empresas a possibilidade de serem reconhecidas como produtoras de software de alta qualidade em seus domínios, tanto em seus mercados internos quanto no mercado internacional, foi desenvolvida a série de normas técnicas ISO/IEC 29110 [ABNT, 2011].

Espera-se que a aplicação da ISO/IEC 29110 possibilite que os gestores de pequenas empresas desenvolvedoras de software respondam positivamente as seguintes perguntas [ABNT, 2011]:

- Sei o que está sendo feito e por quê?
- Posso garantir que cada membro da minha equipe tem o mesmo entendimento do que está sendo feito?
- Tenho controle sobre a integração do que dois ou mais desenvolvedores produzem?
- Posso desenvolver um produto dentro do prazo e do orçamento estabelecidos?
- Posso garantir que o software produzido, tecnicamente, faz o que deveria fazer?
- Posso garantir que o software produzido faz o que o cliente quer que ele faça?

- Quando as mudanças ocorrem – e sei que elas ocorrem – estou preparado para conhecer os seus impactos e para incorporá-las ao trabalho da forma adequada?
- Sei o ponto exato em que o meu projeto de desenvolvimento é encerrado e passa a ser um projeto de manutenção de produto?
- Estou pronto para um aumento de demanda de clientes e produtos?

Respostas positivas às perguntas anteriores podem aumentar a segurança do atendimento aos requisitos dos clientes, a conformidade dos processos aos requisitos da norma e a construção mais rápida de produtos e serviços cada vez melhores.

Além de proporcionar uma certificação que pode ter reconhecimento internacional e facilitar o acesso ao mercado externo, a adoção da série ISO/IEC 29110 por uma VSE pode propiciar outros benefícios, como, por exemplo, o estabelecimento de processos internos de gestão e implementação de software adequados ao seu contexto, aumento da confiança e satisfação dos clientes, maior qualidade do produto ou serviço de software, aumento de patrocínio para a melhoria de processos e diminuição dos riscos de desenvolvimento.

Os processos de ciclo de vida descritos na ISO/IEC 29110, entretanto, não têm intenção de restringir ou desencorajar seu uso em organizações maiores. Podem ser usados pelas VSE tanto ao adquirir e utilizar um sistema de software, quanto ao criá-lo e/ou fornecê-lo para uma terceira parte. Tais processos podem ser aplicados a qualquer nível na estrutura de um sistema de software e a qualquer estágio no ciclo de vida, e não têm intenção de impedir ou desestimular o uso de processos adicionais que as VSE considerem úteis. A Tabela 10 apresenta os processos que compõem o perfil básico da ISO/IEC 29110. Uma organização deve adaptá-lo às suas necessidades e, também, escolher que atividades considerar e como executá-las.

Tabela 10 - Processos da ISO/IEC 29110

Processo	Atividade	Tarefas
Gerenciamento do Projeto	Planejamento do Projeto	Revisar o Escopo do Projeto
		Definir com cliente as instruções de cada entrega
		Identificar as tarefas
		Estabelecer o tempo de duração para cada tarefa
		Identificar e documentar os recursos
		Definir a composição da equipe, definindo papéis e responsabilidades Definir o cronograma estimado de cada tarefa do Projeto Calcular o custo/esforço do Projeto
		Identificar os Riscos do Projeto
		Documentar como será o controle de versão do projeto

Processo	Atividade	Tarefas
		Criar Plano do Projeto integrando o que foi definido anteriormente
		Incluir a descrição do produto, escopo, objetivos e entregas no Plano do Projeto
		Verificar e obter a aprovação do Plano do projeto
		Revisão e aceitação do Projeto
		Estabelecer o Repositório do projeto, usando a estratégia de controle de versão
	Execução do Plano do Projeto	Monitorar a Execução e Criar Relatório de Progresso
		Analisar e Estimar Pedidos de Mudança
		Conduzir Reuniões com Equipe
		Fazer backup de acordo com a Estratégia de Controle de Versão
	Controle e Avaliação do Projeto	Avaliar o progresso do Projeto de acordo com o Plano do Projeto
		Definir ações de correção para os problemas e os riscos identificados. Documentá-las.
		Identificar mudanças nos requisitos e/ou no Plano de Projeto para analisar potenciais riscos e problemas para a conclusão do Projeto
	Fechamento do Projeto	Formalizar a completude do projeto de acordo com as Instruções de Entrega estabelecidas no Plano do Projeto. Conseguir a assinatura do termo de aceite
		Atualizar o repositório do Projeto
Implementação do Software	Início da Implementação do Software	Revisar o Plano do projeto com a Equipe
		Montar/ Atualizar o ambiente de Implementação
	Análise dos requisitos de Software	Definir as tarefas para os membros da equipe de acordo com seus papéis
		Documentar ou atualizar as Especificações de Requisitos.
		Verificar e obter a aprovação do documento de Especificação de Requisitos
		Validar se a especificação de Requisitos satisfaz e atende as expectativas. Os resultados encontrados são documentados em Validação de Resultados
		Documentar a versão inicial do Manual do usuário ou atualizar.
		Verificar e obter a aprovação do manual do usuário, verificar consistência com a Especificação de requisitos
		Incorporar o documento de Especificação de Requisitos e o Manual do usuário na configuração do software na baseline
	Arquitetura de Software e Detalhamento do Design	Definir tarefas para a equipe
		Entender as Especificações de Requisitos
		Documentar ou atualizar o Design de Software. Analisar a Especificação de requisitos para gerar a arquitetura do software. Gerar o documento de Rastreabilidade
		Verificar e obter a aprovação do Design de Software. Os resultados serão documentados no documento de Verificação de resultados e se precisar de mudanças começar um pedido de mudanças
		Criar ou atualizar Casos de Teste e Procedimentos de Testes para testes integrados baseados na Especificação de Requisitos e Design de Software
		Verificar e obter aprovação para os casos de testes e os procedimentos de teste. Documentar em Verificação de resultados
		Atualizar o documento de Rastreabilidade incorporando os Casos de teste e os Procedimentos de Teste
		Incorporar o documento de Design de Software e o de Rastreabilidade as configurações de Software como parte da baseline. Colocar casos de testes e procedimentos de teste no Repositório do Projeto

Processo	Atividade	Tarefas
	Construção do Software	Definir as tarefas para os membros da equipe
		Entender o Design de Software
		Construir ou atualizar os componentes de Software baseado no detalhamento do Design de Software
		Criar ou atualizar os casos de teste de unidade e aplicá-los para ver se os componentes de Software implementam o que foi detalhado no design de software
		Corrigir os defeitos encontrados até que o sucesso dos testes unitários sejam alcançados
		Atualizar a rastreabilidade incorporando os componentes de software construídos ou modificados
		Incorporar os componentes de software e a rastreabilidade nas configurações de software como parte da baseline
	Integração de Software e Testes	Definir as tarefas para os membros da equipe
		Entender casos e procedimentos de testes. Definir/atualizar o ambiente de testes
		Integrar o software usando componentes de software e atualizar Casos e Procedimentos de testes para integração quando necessário
		Realizar testes de software usando Casos e Procedimentos de testes para integração e produção do software e documentar os resultados no "Test Report"
		Corrigir os defeitos encontrados e realizar testes de regressão até o critério de saída ser atingido
		Atualizar o documento de rastreabilidade, se propício
		Documentar o guia de operação do Produto ou atualizar o existente
		Verificar e obter a aprovação do Guia de Operação do Produto. Os resultados encontrados são documentados em Resultados de Verificação e as correções feitas antes do DES aprovar
		Criar a Documentação do Usuário do Software ou atualizar a existente.
		Verificar e obter aprovação da Documentação do Usuário do Software. Os resultados encontrados são documentados em Verificação de resultados e as correções feitas até o documento ser aprovado pelo Cliente
		Incorporar Casos e procedimentos de Testes, Test Report, Guia de Operação do produto e Documentação do Usuário de Software para as configurações de software como parte da baseline
	Entrega do Produto	Definir as tarefas para os membros da equipe
		Entender Configuração de Software
		Criar o documento de Manutenção ou atualizar o existente
		Verificar e obter a aprovação do documento de Manutenção. Os resultados documentados na Verificação de Resultados e as correções são feitas até o documento ser aprovado pelo TL
		Incorporar o Documento de Manutenção como baseline para a configuração de Software
		Realizar entrega de acordo com as Instruções de entrega

2.4 Conclusão

Este capítulo apresentou uma revisão sobre engenharia de software e normas e modelos de qualidade de software.

A ISO/IEC 12207 [ISO/IEC, 2008] contém um conjunto de processos, atividades e tarefas projetado para ser adaptado de acordo com cada projeto de *softwar*e. A estrutura cobre o ciclo de vida do software desde a concepção de ideias até a descontinuação do software. O processo de adaptação consiste na supressão de processos, atividades e tarefas não aplicáveis. As partes envolvidas são responsáveis pela seleção de um modelo de ciclo de vida para o projeto e pelo mapeamento dos processos, atividades e tarefas dentro desse modelo.

O Modelo de Referência MPS para Software (MR-MPS-SW) define níveis de maturidade que são uma combinação entre processos e sua capacidade. A capacidade do processo é a caracterização da habilidade do processo para alcançar os objetivos de negócio, atuais e futuros; estando relacionada com o atendimento aos atributos de processo associados aos processos de cada nível de maturidade.

A norma ISO/IEC 29110 tem como objetivo introduzir conceitos de processos, ciclo de vida e normalização e apresenta perfis de processos adequados a pequenas entidades. Uma vantagem em relação à ISO/IEC 12207 é que a ISO/IEC 29110 já contém um processo de alto nível adaptado para uma pequena entidade. Pode-se entender, dessa forma, que a ISO/IEC 29110 pode ser interpretada como o produto da execução do processo de Adaptação da ISO/IEC 12207 para uma pequena organização de software.

Ao seguir essas normas e modelos é esperado que uma organização consiga elaborar o seu próprio processo para a elaboração de um produto de software. Dessa forma, utilizando estas normas cria-se um padrão para aquela empresa produzir o seu software de acordo com o processo definido e os requisitos dos clientes.

Mas como o foco deste trabalho é o ensino de conceitos relacionados à definição de processos em sala de aula e não em uma organização, é esperado que os conceitos envolvidos sejam absorvidos pelos alunos para o melhor entendimento da proposta do jogo. No entanto, o jogo não deixa de ser uma simulação de como uma organização deverá montar seu processo, com os empecilhos e problemas que podem surgir ao longo do desenvolvimento do produto final.

3. Jogos e Engenharia de Software

3.1 Introdução

Geralmente as grades curriculares dos cursos de graduação preveem o ensino de Engenharia de Software mais ao fim dos cursos. Assim, devido à maior familiaridade com o desenvolvimento de software, espera-se que os alunos estejam mais preparados para o entendimento dos conceitos apresentados. Os jogos baseados em Engenharia de Software têm como objetivo maior a fixação dos conteúdos de forma dinâmica e eficiente. São úteis por apresentar uma outra forma de introduzir os conteúdos aos alunos e também por proporcionar uma maneira diferente de manter o interesse dos alunos no assunto.

O objetivo específico de aprendizagem de um jogo é reforçar conceitos e prover competência na aplicação dos conhecimentos obtidos, atingindo os níveis cognitivos de lembrança, compreensão e aplicação baseando se na taxonomia de objetivos educacionais de Bloom [Anderson, 2001], conforme pode ser vista na Figura 3.

Seja o jogo digital ou não digital (por exemplo, tabuleiros, cartas, interpretativos), todos têm por base passar algum conteúdo. No caso de Engenharia de Software, há exemplos de apoio ao Processo de Medição [Lino, 2007], Gerência de Configuração [Figueiredo *et al.*, 2010] e semelhantes. Apesar de os conteúdos da Engenharia de Software serem abordados por diferentes jogos, não foi encontrado nenhum que englobe todo o conteúdo da Engenharia de Software. Dependendo da metodologia utilizada pelo professor, os alunos podem ser estimulados a competirem de forma saudável entre si, serem beneficiados com pontuações extras pelas atividades, ou outra forma que o professor achar que possa estimular os alunos a aderirem à ideia do jogo.

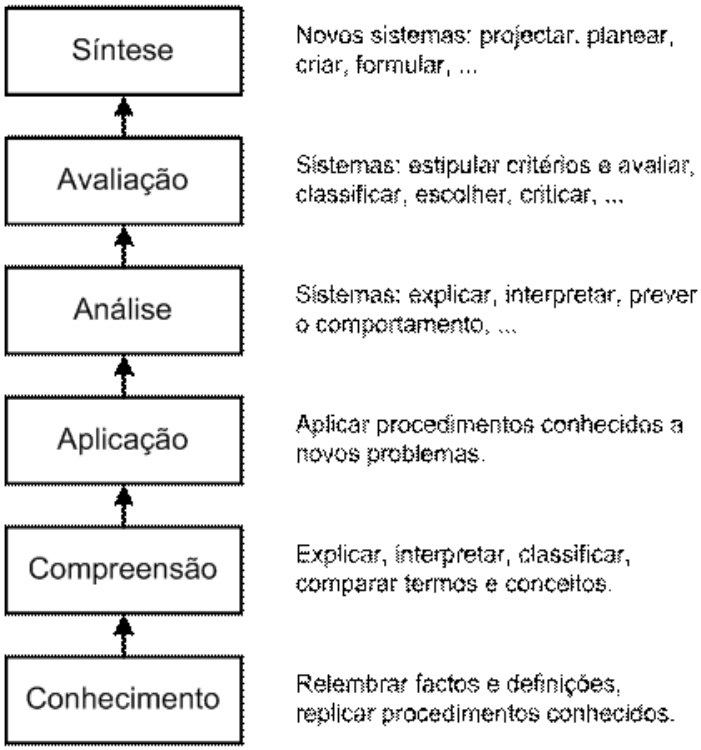


Figura 3 - Taxonomia de Bloom

Em geral, os jogos devem ser aplicados após a matéria específica ser lecionada, pois assim o conteúdo estará em processo de assimilação, e o jogo ajudaria a fixar. Para medir o aprendizado é recomendado que se aplique um questionário sobre a matéria antes e depois do jogo, assim consegue-se avaliar resultados e se eles são significativos o suficiente para indicar que um jogo realmente auxilie o processo de aprendizagem.

A seção a seguir apresenta exemplos de jogos em Engenharia de Software.

3.2 Trabalhos Correlatos

A busca por trabalhos correlatos foi feita de maneira simples, através de artigos selecionados anteriormente à confecção desta proposta, porém a busca se limitou ao cenário nacional. A ideia foi estender e aprimorar essa busca para gerar um entendimento das propostas já existentes de jogos relacionados à Engenharia de Software. Neste trabalho, a análise dos jogos encontrados foi feita de forma a entender qual era o objetivo proposto pelo autor e para gerar um embasamento teórico mais amplo para a comparação com o jogo que está sendo proposto.

Serão abordados o JEES [Figueiredo *et al.*, 2010], A Ilha dos Requisitos [Gonçalves *et al.*, 2010], O Jogo das 7 Falhas [Diniz e Dazzi, 2011] e também uma aplicação de um jogo de RPG (Role Playing Game) para a institucionalização de processos em uma empresa [Suzuki *et al.*, 2011]. Não foi encontrado nenhum trabalho anterior sobre que englobe todos os temas abordados de forma única pelo jogo proposto neste trabalho para ensino de processos de software, sendo assim não há como fazer comparações ou análises de uma ideia existente com a ideia que será proposta.

3.2.1 JEES

O JEES – Jogo de Estratégia para o Ensino de Engenharia de Software [Figueiredo *et al.*, 2010] é uma proposta de jogo que visa transmitir conhecimentos por meio de conceitos e práticas sobre Gerência de Configuração, através de um jogo de tabuleiro com diversos tipos de cartas. O tabuleiro é a área onde o jogador desenvolve o seu projeto, e as cartas contêm o conteúdo do processo de Gerência de Configuração e também de propósito geral. Segundo os autores, o jogo pode ser expandido para outras áreas da Engenharia de Software como Medição ou Garantia da Qualidade.

De acordo com o MR-MPS-SW, a Gerência de Configuração estabelece e mantém a integridade de todos os produtos de trabalho de um processo ou projeto e os disponibiliza a todos os envolvidos, assumindo um papel fundamental de apoio ao controle da evolução [Dart 1991] durante o desenvolvimento e, em especial, durante a manutenção de software.

O jogo tem como objetivo concluir um projeto de software onde as equipes devem entregar as *releases* que estão em uma carta de projeto que cada uma delas recebe. Ele é dividido em turnos específicos e pode contar com um ou mais jogadores. Porém, recomenda-se nesta fase inicial de criação do jogo, pelo menos dois jogadores.

O jogo foi avaliado com o seu uso em três turmas de graduação em Engenharia de Software. O objetivo foi demonstrar como os alunos perceberam o jogo e se conseguiriam entender os conceitos de Gerência de Configuração propostos. Para isso foram elaborados dois questionários de avaliação, um sobre características do jogo em si para avaliar a qualidade do jogo, com perguntas e opções como pode ser visto na Tabela 11.

Tabela 11 - Questionário de avaliação da qualidade do jogo

PERGUNTAS	OPÇÃO 1	OPÇÃO 2	OPÇÃO 3
Após ter jogado JEEES, você diria que:	Gostou	Gostou muito	Não gostou
Você diria que aprendeu com o jogo?	Sim	Não	-
Entre o jogo e uma aula, você prefere:	Aula e jogo	Jogo	Aula
O jogo despertou seu interesse por Gerência de Configuração?	Sim	Não	-

Outro questionário foi elaborado com perguntas sobre o conteúdo do jogo para verificar se o conteúdo proposto foi compreendido pelos alunos, com perguntas como pode ser visto na Tabela 12.

Tabela 12 - Questionário de avaliação do conteúdo de Gerência de Configuração

PERGUNTAS	OPÇÕES DE RESPOSTA
Um dos objetivos principais da Gerência de Configuração é controlar:	A evolução do software A qualidade do software O custo do software
Quais práticas devem ser adotadas para que as equipes de um projeto saibam como trabalhar melhor em grupo paralelamente, evitando problemas como os de merge, por exemplo?	Políticas de Backup Políticas de Engenharia Concorrente Políticas de Configuração
O documento que descreve todo o processo de Gerência de Configuração de um projeto é chamado de:	Estratégia de Gerência de Configuração Política de Gerência de Configuração Plano de Gerência de Configuração
O conjunto de técnicas que descrevem como construir um ambiente de desenvolvimento apoiado por Gerência de Configuração para criar software mais rápido e confiável é chamado de:	Padrões de Gerência de Configuração Treinamento de Gerência de Configuração Técnicas de Desenvolvimento Ágil

Com relação ao primeiro questionário, os resultados foram satisfatórios e a maioria afirmou ter aprendido com o jogo além de o mesmo estimular o interesse por Gerência de Configuração. O segundo questionário também obteve resultados satisfatórios, porém em algumas questões o resultado foi abaixo do esperado, pois estes conteúdos não foram vistos a tempo pelos participantes devido a dinâmica inicial do jogo. Por meio destes questionários os pesquisadores obtiveram *feedback* para poder continuar e melhorar a elaboração do jogo.

3.2.2 A Ilha dos Requisitos

O jogo Ilha dos Requisitos [Gonçalves *et al.*, 2010], por sua vez, tem por foco a Engenharia de Requisitos de Software. É um jogo digital educativo, com interatividade com o usuário para que os estudantes reforcem conceitos e aprofundem seus conhecimentos. Os conteúdos foram inseridos em um contexto de jogo de aventura, onde as ações corretas fazem o personagem avançar, e as erradas atrasam o andamento das ações.

O objetivo do jogo é demonstrar o quanto o mesmo pode apoiar o ensino como uma atividade motivante, e também auxiliar no ensino dos principais conceitos de Engenharia de Requisitos como: (i) o processo de Engenharia de Requisitos; (ii) o papel e responsabilidade do analista de requisitos; e (iii) tipos de requisito. O jogo também tem como finalidade entreter os jogadores através de seu enredo e personagens, contextualizado em ambiente lúdico. Durante uma partida os jogadores deverão enfrentar desafios que contenham o conteúdo abordado na Engenharia de Requisitos, que estão imersos no contexto do jogo, com o fim de deixar menos formal o ensino deste conhecimento.

A Ilha dos Requisitos é um jogo de aventura e estratégia, onde são apresentados diversos desafios, que devem ser resolvidos pelos jogadores, que acabam, também, por ter uma interação indireta com outros jogadores através de um ranking de pontuações que exibe o resultado de todos. O jogo foi aplicado com a participação de alunos de curso superior em Ciências da Computação e do curso de Tecnólogo em Gestão de Tecnologia da Informação de duas instituições de ensino, onde os professores eram do mesmo grupo de pesquisa.

A avaliação do jogo foi feita através de um pré e um pós teste para medir a capacidade nos níveis de entendimento e compreensão. A aplicação do jogo foi realizada entre os dois testes com a finalidade de verificar se o jogo contribuía para o aprendizado. Esta verificação foi feita avaliando os dados destes testes. Também foi aplicado um questionário para a verificação da aceitação do jogo, levando em conta perguntas sobre o objetivo do jogo, sobre a sua aplicação e como ele se apresenta. As perguntas feitas foram:

- Os objetivos de aprendizagem foram atingidos com o jogo?
- O grau de dificuldade do jogo foi adequado para o aprendizado?
- O método de ensino do jogo foi adequado?
- A contextualização apresentada pelo jogo foi adequada?

A partir das respostas foi possível demonstrar que os alunos a maioria dos alunos gostou do jogo e se sentiram estimulados. Além disso, o jogo foi relevante para o aprendizado, agregando conteúdo aos participantes, indicando que a maioria dos jogadores obteve uma melhoria significativa na aprendizagem. A maioria também afirmou que o jogo apresenta de forma adequada a sequência de tópicos, seu método de

ensino e a contextualização apresentada. Os estudantes também tiveram interesse em jogar novamente, apontando melhorias para versões futuras.

3.2.3 Jogo das 7 Falhas

O Jogo das 7 Falhas [Diniz e Dazzi, 2011] tem como proposta ensinar uma característica importante de garantia da qualidade de software, que é o teste de software utilizando a técnica de testes de caixa-preta[1] como estratégia. Dessa forma, segundo os autores, o jogo pretende preencher uma lacuna existente nos cursos de graduação, pois a atividade de testes de software tem sido pouco exploradas. Assim como o jogo citado na subseção anterior, este também é um jogo digital. O jogo consiste em descobrir as falhas existentes nas funcionalidades de um software, onde o jogador deve descobrir as sete falhas existentes em cada funcionalidade testada, correlacionando as falhas com uma classe de equivalência ou um valor-limite.

Com a finalidade de avaliar a qualidade deste jogo, foi realizado um experimento com alunos do curso de Ciência da Computação da UNIVALI, através de aulas sobre teste de software e questionários. Os participantes realizaram o pré-teste e após a realização do mesmo foram divididos em dois grupos: experimental e controle. O grupo experimental utilizou o jogo como forma de aprendizado e após este tarefa realizaram o pós-teste. O grupo de controle ao invés de dispor do jogo realizaram exercícios práticos com o mesmo conteúdo do jogo, acompanhados de outro profissional, que teve a função de orientá-los. Este grupo também realizou o pós-teste. Os pós-testes tiveram como objetivo avaliar o grau de aprendizagem e assimilação do conteúdo após a aplicação do jogo.

De acordo com os dados obtidos com os resultados, o efeito de aprendizagem absoluto entre os grupos experimental e de controle é significativamente diferente, demonstrando uma melhora significativa no efeito de aprendizagem do grupo experimental em relação ao de controle.

Após a realização do experimento, foi aplicado aos participantes um questionário de avaliação do jogo com as seguintes perguntas:

---

[1] O teste de caixa preta não necessita de conhecimento da lógica interna do sistema para se construírem os casos de testes (Perry, 2006). Nessa técnica os casos são construídos a partir de uma especificação do programa (Chen e Poon, 2004).

- "Você achou o jogo agradável?"
- "O jogo é atrativo e conseguiu motivá-lo a executar as tarefas?"
- "Você gostou de jogar o jogo de teste de software?"
- "O que mais você gostou no jogo?"
- "O que menos você gostou no jogo?"
- "Você teria alguma sugestão de melhoria para o jogo?"

De acordo com as respostas obtidas por meio deste questionário, foi possível identificar alguns pontos positivos, como, por exemplo, a interatividade com o aprendizado, e alguns negativos, como a dificuldade de encontrar alguns erros. Também foram sugeridas melhorias que envolviam a criação de erros mais fáceis e listas de testes já realizados. As avaliações realizadas indicaram que o jogo é uma atividade efetiva de ensino, capaz de motivar os alunos e consequentemente, podendo servir como apoio ao instrutor no ensino do teste de caixa-preta.

3.2.4 Utilização de Jogos de RPG (*Role Playing Game*) para Institucionalização de Processos

Este jogo teve por objetivo intensificar os treinamentos nos processos e assuntos acerca de termos e conceitos de maturidade de Software no caso a obtenção do nível 5 do CMMI [Chrissis *et al.*, 2007]. Com esta finalidade, a Spread Systems – Unidade MSA Infor, realizou uma gincana como estratégia de treinamento dos seus colaboradores. Esta gincana seguia o formato de reuniões presenciais, nas quais um membro do GDQ (Gestão da Qualidade) conduzia um jogo de perguntas e respostas.

Através de alguns *feedbacks* recebidos, foi constatado que alguns dos participantes se sentiram pressionados, além de haver o questionamento se este era o método mais adequado de jogos de gincana. Sendo assim, foi feito um aprimoramento no formato dos jogos. Segundo os autores, através de pesquisas acerca dos jogos de educação, idealizou-se a utilização do RPG (*Role Playing Game*), considerando-se o processo dinâmico o qual este jogo proporciona. Foi realizada uma nova gincana com o objetivo de adaptar o formato do jogo RPG, utilizando seu modelo de jogo para a realização na empresa através desta gincana, promovendo o envolvimento efetivo dos colaboradores, através de uma saudável competição entre os envolvidos.

Os colaboradores, de acordo com seus grupos de trabalho de cada projeto de desenvolvimento de Software, foram divididos em equipes homogêneas, e todas participavam individualmente de reuniões. Os papéis a serem desempenhados por cada colaborador foi definido de acordo com a função atual que cada profissional desempenhava nos projetos. Foram escritas histórias em cenários fictícios, por membros da equipe GDQ (Gestão da Qualidade), com o objetivo de simular e relatar o ciclo de vida de um projeto, relatando uma história sequencial e envolvente, abrangendo situações corriqueiras e problemas sobre os quais os participantes seriam direcionados a interagir, interpretar e apresentar as soluções. Dessa forma, proporcionou-se uma total interação entre seus componentes para a execução do trabalho e solução de situações de conflito e verificando o desenvolvimento destas pessoas perante as diversas situações do cotidiano de um projeto. Todos os participantes observavam a execução da ação, constituindo uma oportunidade para o conhecimento em detalhes, das responsabilidades e funções daquele papel e, caso necessário, poderiam complementá-la.

Com a utilização de histórias no formato RPG, os colaboradores vivenciaram o fluxo completo do processo da organização, incluindo as entradas e saídas de cada uma das atividades, mesmo as de responsabilidade de outras funções, garantindo como estratégia o treinamento e proporcionando benefícios à organização e aos próprios colaboradores. Percebeu-se maior motivação para os treinamentos, desenvolvimento da capacidade de trabalhar em equipe, disseminação do conhecimento quanto aos processos organizacionais e um ambiente de trabalho colaborativo. A premiação simbólica ao final de cada gincana configurou uma forma de reconhecimento pelo bom desempenho, demonstrou-se importante na manutenção do clima competitivo entre as equipes.

Como resultado final desta experiência, foi possível identificar oportunidades de melhoria nos processos, problemas recorrentes quanto ao uso dos artefatos, dúvidas ainda existentes em relação aos novos conceitos sobre o processo de maturidade nível 5 do CMMI, assim como a maior assimilação deste conteúdo. O treinamento também ajudou na comunicação entre a área da qualidade e a área de processos organizacionais e demonstrou o comprometimento de todos os envolvidos com o objetivo final.

3.3 Conclusões

De acordo com Thiry *et al.* [2010], jogos educativos podem ser aplicados como um complemento e capacitação de profissionais, considerando que principalmente para o auto-estudo, um jogo é muito mais motivador do que um conjunto de exercícios ou de aulas teóricas. Os jogos têm como objetivo transmitir o conhecimento de maneira lúdica e interativa, onde a participação dos envolvidos atua diretamente no resultado final das escolhas feitas. São cenários onde erros e acertos trazem consequências, mas limitadas ao âmbito do jogo. Assim sendo, podem se tornar ferramentas de grande auxílio na tomada de decisões, já que uma decisão errada, não acarreta em grandes consequências para a empresa.

Foi visto que os jogos apresentados em âmbito acadêmico ou profissional, contribuindo efetivamente para a aprendizagem e treinamento. Estas atividades apresentavam conceitos e conteúdos de forma dinâmica, contribuindo melhor na assimilação, se comparados aos métodos e formas mais tradicionais. Assim sendo, os envolvidos se sentiram mais motivados com este tipo de atividade e demonstraram adesão à proposta e o interesse em jogar novamente.

A ideia inicial do trabalho aqui proposto foi baseada no JEES [Figueiredo *et al.*, 2010], que consiste em ampliar o conhecimento em Engenharia de Software. Ambos apresentam propostas similares, o modelo e confecção são parecidos. Ambos os trabalhos utilizam a ideia de ser um jogo de tabuleiro com cartas e visam ao mesmo objetivo de ensinar conceitos da Engenharia de Software, no entanto o foco específico do jogo apresentado por essa monografia é em Processos de Software. Outros jogos aqui citados foram determinantes para a utilização do método de avaliação usado neste trabalho.

4. ProcSoft – Um Jogo para o Ensino de Processos de Software

4.1 Introdução

O ProcSoft é um jogo educacional com o objetivo de ensinar conceitos relacionados à definição de um processo de software para uma organização utilizando normas e modelos de qualidade de software de maneira informal e descontraída. Além disso, o jogo apresenta conceitos gerais de Engenharia de Software que são relacionados à definição e estrutura de processos de software. A ideia geral é ensinar como organizar um processo de software de maneira mais completa em relação às etapas de um ciclo de vida tendo como base a norma ISO/IEC 29110.

Esta proposta foi elaborada baseando-se no problema de o ensino de Engenharia de Software e por consequência Processos de Software ser muito conteudista, o que costuma trazer alguns problemas como desinteresse do aluno, e também a não absorção do conteúdo completo. O jogo aqui criado tenta dinamizar o ensino deste conteúdo para justamente haver um aprendizado melhor e mais duradouro.

4.2 Descrição do Jogo

O jogo segue o modelo da maioria dos jogos de tabuleiros convencionais, como, por exemplo, o Banco Imobiliário®, e de outros jogos encontrados na literatura sobre Engenharia de Software, como o JEES [Figueiredo *et al.*, 2010], o qual também inspirou a realização deste trabalho. A apresentação do jogo é disposta na forma de um tabuleiro, alguns participantes, cartas, tarefas a serem desenvolvidas conforme sua continuidade. Ele é composto por:

- 28 cartas de papéis;
- 16 cartas de ferramentas;

- 44 cartas de atividades;
- 1 gabarito de processos;
- 20 cartões sorte/revés;
- 4 peões;
- 2 dados;
- 180 notas de dinheiro;
- 1 tabuleiro;
- 60 cartas com perguntas sobre engenharia de software;
- 4 cartilhas com as regras.

O Anexo I apresenta o Formulário utilizado para a avaliação do ProcSoft, a lista de perguntas sobre o conteúdo do jogo e a cartilha de regras do jogo. O Anexo II apresenta os esquemas dos tabuleiros e o gabarito para marcação do resultado. E o Anexo III apresenta as cartas de atividades, as cartas de perguntas, as cartas de atividades, as cartas de papéis, os cartões de Sorte/Revés, gabarito com as atividades previstas para a definição dos processos.

4.2.1 O Tabuleiro

O tabuleiro utiliza atividades de processos sintetizados e selecionados da norma ISO/IEC 29110 distribuídos em 44 (quarenta e quatro) casas, de forma a utilizar as atividades e tarefas mais relevantes da ISO/IEC 29110 e também evitar repetições de mesmo nome. Estas atividade estão divididas em 4 (quatro) grupos e cores:

- Atividades executadas pelo Gerente – Roxo;
- Atividades executadas pelo Analista – Rosa;
- Atividades executadas pelo Desenvolvedor – Verde;
- Atividades executadas pelo Testador – Laranja.

No tabuleiro também são encontradas quatro casas com conteúdos diferentes de atividades de processo. Estas casas contêm sorte/revés, $$ (casa que indica “Dinheiro”), papel e ferramentas. Além disso, ao parar em cada casa de atividades a equipe

participante deverá responder uma das 60 (sessenta) perguntas relacionadas a Engenharia de Software.

O objetivo destas casas, além de dar dinâmica ao jogo, é criar facilidades e dificuldades aos jogadores, mas também demonstrar necessidades e acontecimentos que podem ocorrer durante a idealização dos processos de software da empresa para a confecção de um produto final.

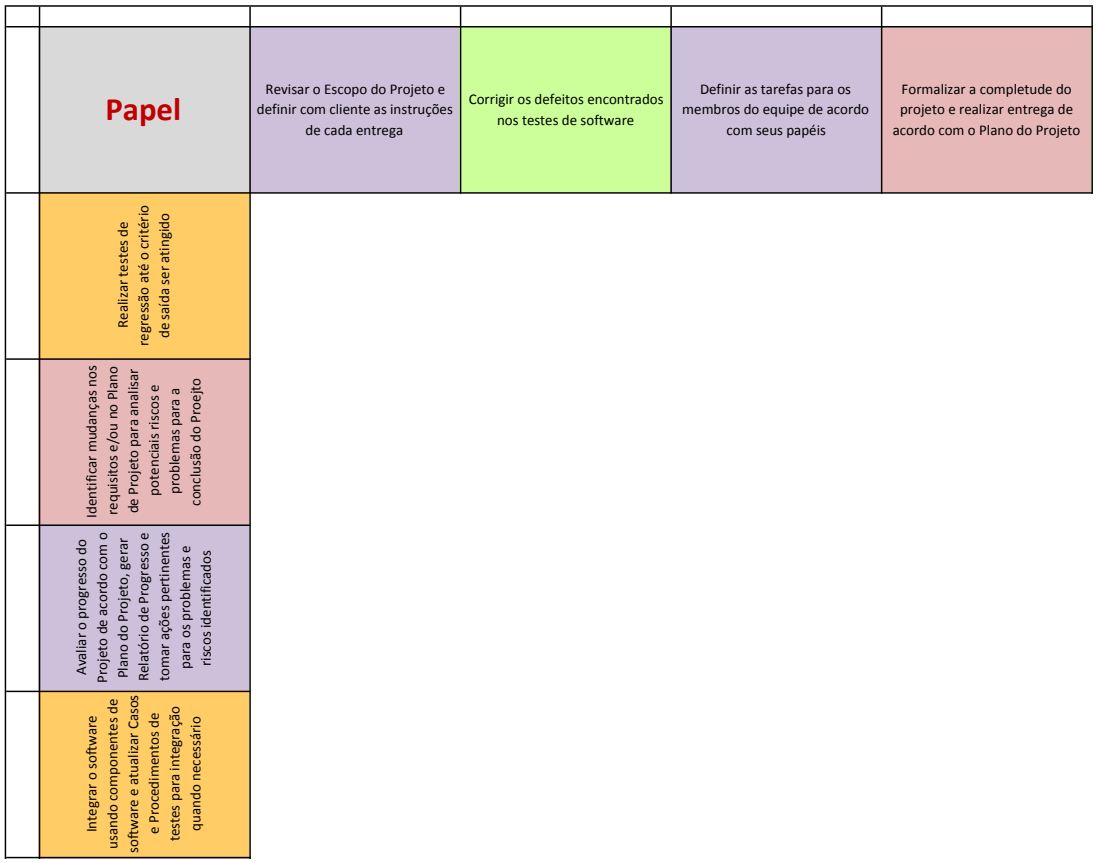


Figura 4 – Parte do Tabuleiro

4.2.2 Atividades

As Atividades representam tarefas existentes na norma ISO/IEC 29110 [ABNT, 2011] para a criação de processos. Estas tarefas foram selecionadas e dispostas no tabuleiro de forma aleatória, porém elas pertencem a 4 (quatro) tipos. Os tipos são:

- Planejamento;
- Execução e Monitoração;
- Análise e Projeto;
- Codificação/Testes/Entrega.

Toda vez que o peão parar em um espaço de Atividade, o jogador poderá comprá-la pelo preço de $100. Mas para isso deverá responder a uma Pergunta corretamente, senão o valor subirá para $200. Ao comprar, você receberá o título daquela atividade. Se não possuir a Ferramenta adequada para a Atividade será cobrado $50,00 a mais. Ao término do jogo, se não possuir a carta de Ferramenta correspondente à Atividade, sua pontuação será reduzida à metade. Caso tenha a carta de Papel sênior correspondente à Atividade ganhará 50 pontos por Atividade correspondente.

As equipes deverão identificar as Atividades corretamente ao tipo em que ela pertence. No jogo, as Atividades não precisam ser colocadas na ordem dentro do tipo ao que ela pertence. A Figura 5 apresenta um exemplo de carta de Atividade.

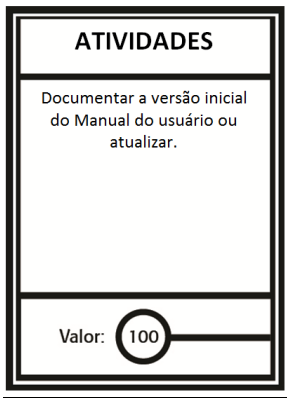


Figura 5 – Exemplo de Carta de Atividades

4.2.3 Os Papéis

Os Papéis servem para representar cargos e responsabilidades que existem em uma empresa real. São cargos essenciais em uma equipe para que possam executar as atividades de Engenharia de Software definidas de acordo com o processo estabelecido pela empresa e também consigam gerenciar o projeto ao longo de sua duração até a obtenção do resultado final.

Antes de o jogo começar, os grupos deverão definir quem interpretará cada papel no grupo: Gerente, Analista, Desenvolvedor e Testador. Na primeira rodada as equipes poderão comprar 2 papéis. Os papéis funcionam como critério de capacidade para poder avançar no jogo, uma vez que algumas ferramentas só poderão ser manipuladas por alguém que tenha um determinado papel e de acordo com seu nível (Júnior, Sênior). A equipe só poderá comprar uma carta de papel Sênior se já possuir a de Júnior. A carta de Gerente só possui apenas um nível (equivalente ao Sênior). Caso uma equipe tenha

menos pessoas que os papéis previstos no jogo, alguns membros da equipe deverão acumular papéis.

O grupo poderá escolher se quer promover um dos papéis toda vez que passar pela casa de papéis (casa inicial do jogo). Para promover, é preciso pagar um valor especificado na carta de papel. Cada papel possui uma bonificação diferente e dá acesso a mais ferramentas, podendo assim acumular mais pontos. O responsável, por representar a equipe no tabuleiro, será o que estiver com o papel de Gerente, caso a equipe não tenha comprado este papel, um membro será escolhido pela equipe para representá-la. A Figura 6 apresenta um exemplo de carta de Papéis. Os papéis, títulos e valores são apresentados na Tabela 13.

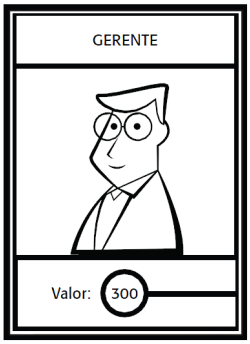


Figura 6 – Carta de Papéis

Tabela 13 – Valor das Cartas de Papéis

Papel	Tipo 1	Valor 1	Tipo 2	Valor 2
Gerente	-	-	-	$300,00
Analista	Júnior	$100,00	Sênior	$300,00
Desenvolvedor	Júnior	$100,00	Sênior	$300,00
Testador	Júnior	$100,00	Sênior	$300,00

4.2.3 Ferramentas

As Ferramentas são cartas que ajudam a realizar as tarefas que se encontram no tabuleiro. Elas são de diferentes tipos: Gerenciamento, Análise e Projeto, Codificação e Testes. Elas podem ser adquiridas ao longo do jogo para auxiliar no processo. Algumas ferramentas só podem ser utilizadas por determinados papéis (Gerente, Analista, Desenvolvedor ou Testador) e níveis (Júnior, Sênior ou Uso Geral). Elas variam de preço de acordo com a sua qualidade/posição no mercado/etc. Você poderá adquirir ferramentas sempre que passar pela casa Ferramentas no tabuleiro. A Figura 7 apresenta

um exemplo de carta de Ferramentas. Os valores das ferramentas são apresentados na Tabela 14.

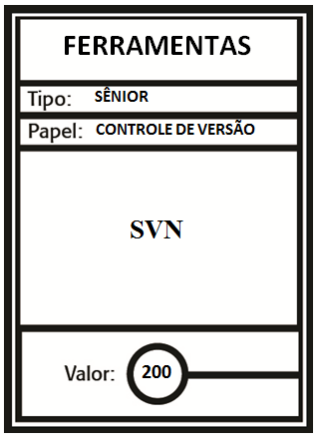


Figura 7 – Carta de Ferramentas

Tabela 14 – Valor das Cartas de Ferramentas

Tipo	Valor
Uso Geral	$100,00
Júnior	$150,00
Sênior	$200,00

4.2.4 Cartas de Sorte/Revés

As equipes deveram tirar Sorte/Revés sempre que pararem na casa de mesmo nome. Uma carta deve ser tirada e a ordem executada imediatamente. Depois, a carta deverá ser devolvida para baixo da pilha. Toda vez que um peão adversário parar em uma Atividade que outra equipe possui, a equipe também deverá retirar uma carta de Sorte/Revés. Caso dê sorte o bônus será revertido para quem possui a Atividade, caso dê Revés a penalidade será para a equipe que parou na Atividade de outra equipe (nesse caso o dinheiro será dado ao banco). A Figura 8 apresenta exemplo de Carta de Sorte/Revés.

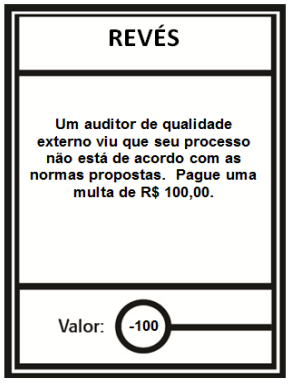


Figura 8 – Carta de Papéis

4.2.5 A Casa $$

A casa $$ (Dinheiro) é utilizada como uma casa bônus, pois cada vez que um peão alcançar esta casa, a equipe receberá uma bonificação, exceto na primeira rodada. O objetivo desta casa é poder auxiliar as equipes a comprarem e evoluírem atividades, ferramentas e papéis ao longo do jogo.

4.2.6 Perguntas

As cartas de perguntas contêm 60 perguntas de diversas áreas da Engenharia de Software. O objetivo principal é ensinar uma parte do conteúdo da disciplina de forma dinâmica e divertida e atrelar este conhecimento ao jogo, utilizando-as de maneira a incrementar a jogabilidade.

Antes de poder comprar uma carta de um determinado tipo, é necessário responder uma pergunta sobre Engenharia de Software. Se a equipe souber responder, paga o valor de face. Não sabendo responder, paga o dobro pela atividade. As perguntas podem ter respostas diferentes do gabarito desde que as equipes participantes concordem que a resposta dada também é válida. A Figura 9 apresenta um exemplo de carta de Perguntas. Exemplos de perguntas incluem:

- O que são modelos de maturidade de software? Cite 2.
- O que é um projeto de software?
- O que são teste de caixa preta?
- O que é engenharia de software?
- O que é processo de software?

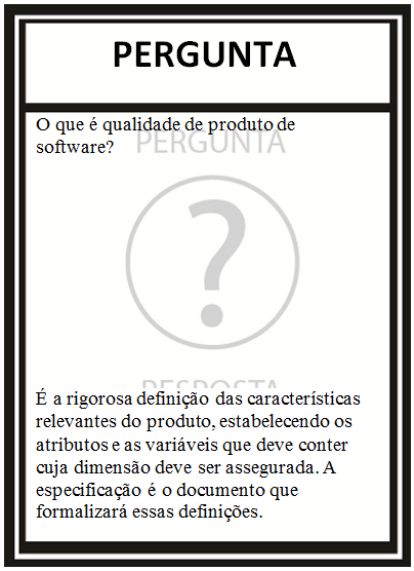


Figura 9 - Carta de Perguntas

4.2.7 Cartão e Gabarito de Processos

Inicialmente cada equipe ganha uma cartela para organizar as atividades adquiridas ao longo do jogo. Nesta cartela estará apenas o tipo da Atividade adquirida e assim a equipe deverá coloca-la no local correto, isto poderá trazer bonificações ou deduções às equipes dependendo se elas acertarem ou errarem. O Gabarito de processos é dado ao término do jogo para que essa checagem possa ser feita e assim cada equipe poder avaliar o seu desempenho nesta tarefa.

4.3 Dinâmica do Jogo

Quem começa o jogo: Joga se um dado, e o grupo com o maior número inicia, seguido pelo que tirou o segundo maior e, assim, sucessivamente.

Começo do jogo: A primeira equipe lança os dados e, conforme o número de pontos obtidos, avança o seu peão no sentido horário para o espaço atingido. Em um só espaço podem parar vários piões ao mesmo tempo.

Se cair em uma tarefa é dada a opção de comprá-la ou não, caso queira compra-la é necessário responder uma pergunta feita pela outra equipe, em caso de resposta correta a equipe adquire a carta e deverá anotá-lo em seu cartão de processos.

Tirando uma dupla (2 e 2, 3 e 3, etc.) a equipe tem direito a novo lançamento.

Término do Jogo: O jogo termina após 60 minutos (ou outra duração acordada entre as equipes). Quem conseguir encaixar o maior número de atividades coerentes ao processo (baseados na ISO/IEC 29110) nas fases corretas, utilizando da melhor forma os papéis e as ferramentas para conseguir uma maior bonificação, será declarado como vencedor. Para isso, deve-se contabilizar:

- Para cada atividade para qual a equipe tenha adquirido uma ferramenta para executá-la, deve-se contar o valor da atividade de face ($100).
- Para cada atividade para qual a equipe não tenha adquirido uma ferramenta para executá-la, deve-se contar o valor da atividade de face ($50).
- Se a equipe tiver algum papel Sênior, ganha-se uma bonificação de $50,00 para as atividades que podem ser desempenhadas por aquele papel.
- Se a equipe tiver alguma ferramenta Sênior, ganha-se uma bonificação de $50,00 para as atividades para a qual aquela ferramenta é necessária.
- Para cada atividade associada a uma fase incorreta, há uma penalidade de $50,00.

4.4 Aplicação do Jogo

A fim de avaliar a primeira versão do ProcSoft foi realizada uma aplicação do jogo em uma turma da disciplina Processos de Software do curso de Sistemas de Informação da Universidade Federal do Estado do Rio de Janeiro (UNIRIO). Esta turma no momento de sua aplicação possuía 13 (treze) alunos. Eles foram divididos em 4 equipes, sendo 3 equipes com 3 alunos e uma com 4. Os alunos escolheram a que grupo pertenceriam.

O objetivo dessa divisão foi estimular a cooperação entre os membros de cada equipe, cabendo a eles decidir que papeis cada assumiria, simular um ambiente competitivo onde decisões e respostas erradas prejudicam a equipe e beneficiam a concorrente e decisões e respostas corretas auxiliam a equipe a prosperar dentro do jogo.

Para verificar se o jogo pode ser útil e motivador para os alunos, era necessário avaliar a percepção em relação ao jogo e também se foi possível aprender algum

conteúdo relacionado à Engenharia de Software, para isto foi elaborado um formulário dividido em duas partes. A primeira parte possui (Tabelas 15 e 16) 9 (nove) questões acerca da opinião dos alunos participante sobre o jogo, sendo 6 de múltipla escolha e 3 discursivas. Na segunda parte (Tabela 17) estão perguntas relacionadas ao quanto o jogo conseguiu transmitir conhecimentos sobre Engenharia e Processos de Software em geral e a norma ISO/IEC 29110, em particular.

Tabela 15 – Questões de múltipla escolha do questionário sobre o ProcSoft

#	Questão	Opção 1	Opção 2	Opção 3
1	Você gostou de jogar o ProcSoft?	Sim	Não	Indiferente
2	Você achou a jogabilidade complicada?	Sim	Não	Indiferente
3	Você diria que aprendeu com o jogo?	Sim	Não	Indiferente
4	Uma versão digital do jogo atrairia mais sua atenção?	Sim	Não	Indiferente
5	Você jogaria o ProcSoft de novo?	Sim	Não	Indiferente
6	Entre o jogo e uma aula, você prefere:	Aula	Jogo	Aula e Jogo

Tabela 16 – Questões discursivas do questionário sobre o ProcSoft

Questões
O que mais você gostou no jogo?
O que menos você gostou no jogo?
Você tem alguma sugestão de melhoria para o jogo?

Tabela 17 – Questões sobre o conteúdo abordado pelo jogo ProcSoft

Questões
Como uma empresa de software pode utilizar a norma ISO/IEC 29110?
Quais elementos mínimos são necessários para se definir um Processo de Software
Que atividades, em geral, ficam a cargo do: • Gerente de Projeto? • Analista? • Desenvolvedor? • Testador?
Você acha que a estrutura do processo formado ao longo do jogo é adequada para uma empresa pequena? Por quê?
Você acha que as perguntas de sorte/revés são representativas para uma empresa de software? Por quê?
Você acha que as perguntas de Engenharia de Software estão de alguma forma ligadas às atividades que o processo apresenta? Comente se achar necessário

A aplicação do jogo juntamente com o questionário teve duração de 2 horas na turma, pois esse era o tempo de duração da aula em que ele foi ministrado. O tempo foi dividido em 10 minutos de apresentação, 80 minutos jogando e 20 minutos respondendo os questionários. A duração foi de acordo com o tempo disponibilizado pelo professor. O tempo planejado inicialmente para jogo foi de 60 minutos, mas devido a interrupções para esclarecimentos sobre o jogo o tempo foi estendido.

Durante o jogo foi possível perceber o engajamento das equipes em entenderem a proposta, e se esforçarem para responder as perguntas sobre Engenharia de Software e fazerem as escolhas que mais beneficiariam as suas empresas fictícias e, também ao escolher ferramentas e papéis adequados às situações que apareciam, sempre levando em consideração a concorrência e ameaça das outras equipes e suas respectivas empresas fictícias.

As Figuras 10 e 11 mostram os resultados obtidos nas questões de múltipla escolha da primeira parte do formulário de avaliação do jogo:

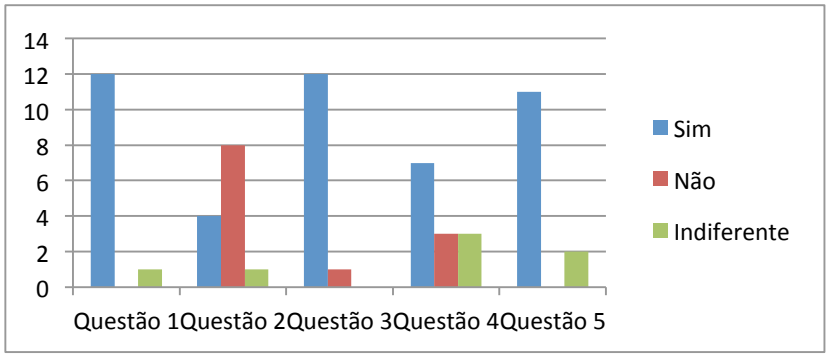


Figura 10 – Respostas das questões (1 a 5) de múltipla escolha do questionário sobre o ProcSoft

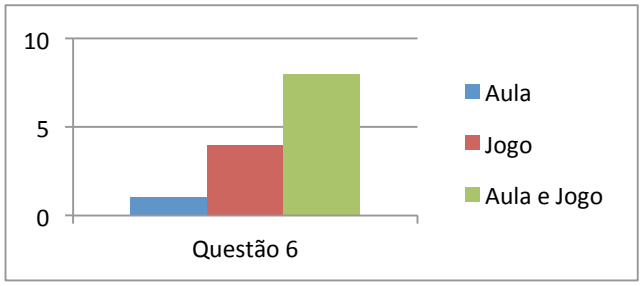


Figura 11 – Resposta da questão 6 (Aula x Jogo) de múltipla escolha do questionário sobre o ProcSoft

Estas questões tinham como objetivo avaliar o jogo em si. Os resultados indicaram que a maior parte dos alunos gostaram de jogar o ProcSoft. Não havendo nenhuma resposta negativa quanto a isso. Apenas uma opinião que se mostrou indiferente ao jogo em si.

Por sua vez, como ainda é uma versão inicial, o jogo ainda não teve uma jogabilidade muito clara para alguns alunos. Mesmo que mais da metade não tenha achado a jogabilidade complicada, o número é significativo para rever alguns aspectos do jogo e deixá-lo mais claro para os jogadores.

Quanto ao aprendizado o retorno foi satisfatório, tendo apenas uma resposta negativa quanto à capacidade de aprender com o jogo. As outras respostas foram positivas, demonstrando que o jogo agregou conhecimento àqueles que participaram da atividade.

Entre o jogo e aula, boa parte dos alunos (8 alunos) ainda acredita que o melhor método é manter os dois, pois a aula traz o conhecimento necessário para o jogo e o jogo os fixa, porém havendo dúvidas, um professor pode explicar melhor um determinado assunto ou expandir uma vez que o jogo nem sempre pode abordar todos os conteúdos.

A ideia de uma versão digital foi considerada positiva, pois poderia dar mais dinamismo ao jogo e também criar um sistema online para isso, ou desktop para jogar sozinho contra outras máquinas simulando equipes ou em grupo. 8 alunos se mostraram interessados em uma versão digital do jogo. Outros 4 se acham que não é necessária uma versão assim do jogo, e um aluno é indiferente se o jogo atrairia mais a atenção sendo digital.

O interesse se o ProcSoft seria um jogo o qual os estudantes sentiriam vontade de jogar novamente também foi avaliado. A grande maioria (11 alunos) se interessou em repetir a experiência, o que foi considerado um bom sinal, pois indica que a experiência foi significante para os alunos.

As questões discursivas sobre o jogo tratam sobre a opinião do aluno em tópicos como aquilo que ele mais gostou no jogo, o que menos gostou e melhorias para o jogo como um todo. Abaixo serão mostradas as perguntas e essas opiniões:

"O que você mais gostou no jogo?"

- A abordagem descontraída para revisar o conteúdo das aulas.
- Sorte/Revés.
- Das pergunta sobre PSW (Processos de Software) e demais áreas.

- Integrar a disciplina de Engenharia de Software ao Banco Imobiliário®, tornando o aprendizado mais divertido.
- Perceber o que todos os estudantes entendem sobre um assunto.
- Apoia o aprendizado dos conceitos da disciplina através da competição entre os grupos.
- A dinâmica de perguntas e respostas possibilitando um aprendizado sem a rigidez de uma aula.
- As perguntas.
- Do fato de jogar e aprender ao mesmo tempo.
- A possibilidade de aprender conceitos de uma maneira mais produtiva.
- A maneira como a matéria foi passada.
- A temática e interação com outras equipes/jogadores.
- As discussões para que os grupos tentassem chegar a uma resposta.

"O que menos você gostou no jogo?"

- Ficar em pé. E organizar as atividades no final (já que elas não eram coloridas).
- Subjetividade das perguntas/respostas.
- Algumas perguntas tinham respostas grandes demais e fica complicado de responder.
- Achei que as perguntas precisam ser melhor elaboradas.
- Algumas perguntas e respostas estavam um pouco confusas.
- Não ter cores nos cartões de atividades.
- As respostas das perguntas serem julgadas pelo próprio aluno, pois não sabemos definir se a resposta é adequada ou não.
- Não achei pontos negativos.
- Perguntas ambíguas.
- Algumas perguntas estavam muito amplas, o que dava margem a várias respostas.
- Algumas perguntas subjetivas, muito abrangentes.
- Achei a regra de papeis júnior/sênior e ferramentas uso geral/sênior um pouco complicadas.
- A ambiguidade de algumas perguntas e respostas.

"Você tem alguma sugestão de melhoria para o jogo?"

- Resolver as ambiguidades e definir as regras mais claramente.
- O jogo precisa ter um juiz estabelecido para decidir se a resposta está correta.
- Melhor definição das regras, numeração em todas as cartas. Seria interessante ao passar por casa como "papel" e "ferramentas" e "dinheiro" o participante escolher se quer comprar uma ou se deseja comprar a atividade. Perguntas indicando matéria referenciada (GPInfo, PSW, etc votação para definir se é acerto ou não (1 voto por equipe)).
- Melhorar as perguntas para que fique claro a intenção da resposta.
- O aplicador do jogo, no caso o banco, deveria fornecer as resposta de certo ou errado, pois ele escreveu respostas para as perguntas e, como as respostas dadas geralmente não são iguais, ele definiria.
- Detalhar um pouco mais certas perguntas/respostas, para que haja um julgamento um pouco melhor das respostas certas/erradas.
- Uma melhor identificação das cartas das atividades.
- Melhorar os textos de forma geral.
- Deveria haver uma forma mais criteriosa para determinar se a s respostas estão corretas.
- Melhorar as questões, criar uma versão digital.
- As perguntas poderiam ser objetivas e uma melhor identificação nas cartas através do uso de cores.
- Incluir o tema da pergunta e não exibir a categoria (cor) da atividade, transformando sua descoberta em ponto? Da pergunta (e "preço").
- Tornar as cartas das atividades coloridas.

De formal geral, percebe-se a necessidade de melhorar as perguntas de engenharia de software do jogo para criar um padrão para as respostas e também a identificação das cartas para simplificar a jogabilidade, porque isto foram os pontos que concentraram a maioria das respostas sobre o que não agradou no jogo e sobre as melhorias sugeridas.

Sobre os pontos positivos, foi verificado que uma abordagem dinâmica para o ensino de Processos de Software acabou agradando aos alunos, e também a possibilidade de trabalhar em grupo simulando uma situação de equipe na vida real. E também poderem verificar o conhecimento que eles e seus colegas de turma possuíam durante o jogo.

As respostas das perguntas sobre os conteúdos também foram avaliadas. Foram 6 perguntas com o objetivo de verificar se a aplicação do jogo foi eficiente para o entendimento e solução delas. Abaixo seguem as perguntas e a avaliação sobre elas:

"Como uma empresa de software pode utilizar a norma ISO/IEC 29110?"

Para esta pergunta, foi avaliado que quase metade dos alunos não conseguiu captar a intenção da norma deixando a resposta da pergunta em branco, com isto é necessário reavaliar como está sendo passado o objetivo da mesma.

"Quais elementos mínimos são necessários para se definir um Processo de Software?"

Era esperado que os alunos respondessem sobre a definição de papéis, atividades e ferramentas para a definição de um Processo de Software. Quanto a esta pergunta, foi verificado que quase todos os alunos entenderam como é definido um Processo de Software, havendo a maioria deles respondido de forma adequada a esta questão.

"Que atividades, em geral, ficam a cargo do:

- *Gerente do Projeto?*
- *Analista?*
- *Desenvolvedor?*
- *Testador?"*

Para essa pergunta, foi visto que todos os alunos conseguiram entender para que cargo as atividades dos Processos de Software são delegadas. Todos responderam de forma correta, mas nem sempre completa a esta questão.

"Você acha que a estrutura do processo formado ao longo do jogo é adequada para uma empresa pequena? Por quê?"

Nesta pergunta foi verificado que a compreensão da maior parte dos alunos foi boa quanto à estrutura de um processo. Que nem sempre ele como um todo será adequado a uma empresa e para isto ele deverá ser adaptado para atender as necessidades de cada organização.

"Você acha que as perguntas de sorte/revés são representativas para uma empresa de software? Por quê?"

Para a maioria dos alunos as situações apresentadas nestas cartas simularam eventos que poderiam ocorrer na realidade em uma empresa. Estas situações, segundo eles, poderiam prejudicar o andamento do projeto e acarretar prejuízos para empresa, ou beneficiar o projeto, gerando ganhos.

"Você acha que as perguntas Engenharia de Software estão de alguma forma ligadas às atividades que o processo apresenta? Comente se achar necessário."

Os alunos em sua maioria responderam 'sim' a este questionamento, pois estas perguntas representam os conhecimentos corretos necessários para a aplicação dos processos e permitem revisar os conceitos das atividades. Porém, é preciso tratar alguns termos ambíguos e genéricos nestas perguntas.

4.5 Conclusão

Com a criação e a aplicação do ProcSoft foi possível avaliar como jogos educacionais podem ser utilizados para ampliar a experiência de aprendizado, indo além da mostra tradicional de conteúdo e trazendo novas experiências para os alunos entrarem em contato com o conteúdo apresentado.

O jogo envolveu os alunos na atividade estimulando-os. A aplicação do questionário pôde mostrar resultados significativos da experiência. Estes resultados mostram o interesse por jogos e por mais dinamismo no método tradicional de ensino. Também demonstram que é possível o aprendizado e a retenção de conhecimentos através deste tipo de atividade.

Jogos educativos podem ser aplicados como um complemento à capacitação dos alunos, considerando através dos resultados do questionário, que a combinação aula e jogo se mostram mais motivador ao aluno do que o conjunto de exercícios e aulas teóricas.

5. Conclusão e Trabalhos Futuros

5.1 Principais Contribuições

A principal contribuição desse trabalho foi a definição do ProcSoft, que tem como objetivo usar jogos educacionais para o ensino de Processos de Software. Neste trabalho foi descrito como o jogo foi criado e a sua aplicação em uma turma da disciplina de Processos de Software do Bacharelado de Sistemas de Informação da UNIRIO a fim de obter resultados para avaliar a sua criação.

O foco do jogo foi o ensino de Processo de Software utilizando a norma ISO/IEC 29110 como base para mostrar como montar um processo, também foram criadas perguntas gerais sobre Engenharia de Software para ampliar o jogo e avaliar outros conhecimentos possuídos pelos alunos.

A avaliação da versão inicial do jogo mostrou que os alunos foram receptivos ao mesmo. Através dos questionários foi possível perceber que a experiência foi positiva, tanto na forma de aplicação do jogo quanto nos resultados obtidos para a demonstração de retenção de conhecimento inspirando o aprimoramento desde trabalho.

5.2 Limitações do Projeto

Uma limitação da avaliação do jogo apresentada nessa monografia é o fato de que ele foi aplicado em apenas uma turma com apenas 13 alunos, em uma só instituição de ensino. Desta forma ainda é necessário ampliar o conjunto de testes do jogo para a verificação dos resultado sobre a aprendizagem e também sobre a sua avaliação.

Outro fator limitante foi a disparidade de conhecimento entre alunos, que foi tratada ao fazê-los trabalhar em grupo para haver uma troca de aprendizagem entre os membros. Isto evitou grandes diferenças ao longo do jogo, deixando a partida equilibrada entre os participantes.

5.3 Trabalhos Futuros

Futuramente o ProcSoft deverá ser revisado e avaliado novamente, para corrigir defeitos como as perguntas ambíguas, a colocação de cores nas cartas e definições de respostas padrão. Também deverá ser revisto o tabuleiro, verificando se as atividades da norma ISO/IEC 29110 são a mais adequada para o jogo. Outra questão é a definição do tempo limite do jogo que deverá ser melhor avaliado, uma vez que atualmente o tempo pode variar de acordo com quem aplica o jogo.

Outro trabalho futuro é a criação de uma versão digital do jogo. Esta versão pode ser online ou desktop. A versão online poderia ser jogada entre várias pessoas diferentes, formando até 4 equipes de 4 componentes para se enfrentarem ao longo do jogo. A versão desktop por sua vez permitiria ao jogador jogar sozinho sendo às equipes, tanto a dele quanto as adversárias, completadas por *bots* (computadores usados como personagens). Assim ele teria a experiência de jogar em casa quando quisesse e podendo aprender o conteúdo e simulando o jogo sem precisar de outras pessoas.

Referências Bibliográficas

ABNT – Associação Brasileira de Normas Técnicas. Guia de implementação: Desenvolvimento de softwares para pequenas organizações [recurso eletrônico] / Associação Brasileira de Normas Técnicas, Serviço Brasileiro de Apoio às Micro e Pequenas Empresas. – Rio de Janeiro: ABNT; SEBRAE, 2012.

Anderson, L. W. "Rethinking Bloom's Taxonomy: implication for testing and assessment". Columbia: University of South Carolina, 1999. (Report n. MF01/PC01).

Brownfield, S.; Vik, G. Teaching basic skills with computer games. Training and Developmental Journal, v. 37, n. 2, p. 52-56, 1983.

Ricci, K. E. The use of computer-based videogames in knowledge acquisition and retention. Journal of Interactive Instruction Development, n. 7, v. 1, p. 17-22, 1994

Chen, T. Y.; Poon, P. L. Experience with teaching black-box testing in a computer science/software engineering curriculum. IEEE Transactions on Education, v. 47, n. 01, p. 42-50, 2004.

Chrissis, M.B., Konrad, M. e Shrum, Sw. (2007) "CMMI: Guidelines for Process Integration and Product Improvement". Addison-Wesley.

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

Dempsey, J.; Gilley, B.; Rasmussen, W. (1993) "Since Malone's theory of intrinsically motivating instruction: What's the score in the gaming literature?" Journal of Educational Technology Systems, v22 n2 p173-83 ISSN-0047-2395.

Dempsey, J. V.; Rasmussen, K.; Lucassen, B. Instructional gaming: implications for instructional technology. Paper presented at the Annual Meeting of Association for Educational Communications and Technology, Nashville, 1994.

Diniz, Lucio L. and Dazzi, Rudimar L. S. (2011). Jogo Digital para o Apoio ao Ensino do Teste de Caixa-Preta. In *X Simpósio Brasileiro de Qualidade de Software*, pages 231-245. Mestrado em Computação Aplicada – Universidade do Vale do Itajaí (UNIVALI).

Figueiredo, K., Ferreira, J., Murta, L. and Clua, E. (2010). Um Jogo de Estratégia de Gerência de Configuração. In *III Fórum de Educação em Engenharia de Software*, pages 1-8. Instituto de Computação – Universidade Federal Fluminense (UFF).

Fuggetta, A., Software Process: A Roadmap, In: Proc. of The Future of Software Engineering, ICSE'2000, Limerick, Ireland, 2000, pp. 25-34.

Gonçalves, Rafael Q., Thiry, M. and Zoucas A. (2010). Avaliação da Aprendizagem em Experimentos com Jogo Educativo de Engenharia de Requisitos. In *X Simpósio Brasileiro de Qualidade de Software*, pages 215-229. Ciência da Computação – Universidade do Vale do Itajaí (UNIVALI).

Griffiths, M. D. The educational benefits of videogames. Education and Health, v. 20, n. 3, p. 47-51, 2002.

IEEE, *IEEE Standarts Collection: Software Engineering*, IEEE Standard 610.12 – 1990, 1993

ISO/IEC – International Organization for Standardization/ International Electrotechnical Comission. ISO/IEC 12207 – Systems and Software Engineering – Software Life Cycle Processes, 2nd edition, 2008.

Lino, J. I. (2007) "Proposta de um Jogo Educacional para a área de Medição e Análise de Software". Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) – Universidade Federal de Santa Catarina, Florianópolis.

Malone, T. C. Guidelines for designing educational computer programs. Childhood Education, v. 59, n. 4, p. 241-47, 1983.

Malone, T. C. What makes things fun to learn? heuristics for designing instructional computer games. Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems, p. 162-169, 1980.

Naur P., and B. Randall (eds), *Software Engineering: A Report on a Conference Sponsored by the NATO Science Committee*, NATO, 1969

Perry, W. E. Effective methods for software testing. Indianápolis, Indiana: Third Edition, Wiley, 2006.

PRESSMAN, Roger S. Engenharia de software. 6ª ed. Porto Alegre: Bookman, 2011.

SOFTEX. MR-MPS - Melhoria de Processo do Software Brasileiro, Guia Geral: Versão 2009. Brasília: Softex, 2009.

Suzuki, Cintya C. C., Xavier, R. V., Romeu, P. N., Moreira, L. V., Freitas, R. L., Moreira, S. C., Junior, Pedro L. R. L. (2011). A utilização de Jogos em Treinamentos e Ações de Institucionalização de Processos. In *X Simpósio Brasileiro de Qualidade de Software, p. 401-408.*

Tarouco, L., Roland, L., Fabre, M. e Konrath M. (2004) "Jogos Educacionais", In: Revista de Novas Tecnologias na Educação, UFRGS, Porto Alegre, V. 2 Nº 1.

Thiry, M., Zoucas, A., Goncalves, R. Q. and Salviano C. "Aplicação de jogos educativos para aprendizagem em melhoria de processo e Engenharia de Software ". In: Workshop Anual do MPS, 2010, Campinas, SP. Anais do VI WAMPS 2010. Campinas, SP : Softex, 2010. v. 1. p. 118-127.

Yee, N. The labor of fun: how video games blur the boundaries of work and play. Games and Culture, v. 1, p. 68-71, 2006.

ANEXO I

Regras Como Jogar o ProcSoft

CONTEÚDO:

- 28 CARTAS DE PAPÉIS
- 16 CARTAS DE FERRAMENTAS
- 44 CARTAS DE ATIVIDADES
- 1 GABARITO DE PROCESSOS
- 20 CARTÕES SORTE/REVÉS
- 4 PEÕES
- 2 DADOS
- 180 NOTAS
- 1 TABULEIRO
- 60 CARTAS COM PERGUNTAS SOBRE ENGENHARIA DE SOFTWARE
- 4 CARTILHAS COM AS REGRAS

TABULEIRO: O tabuleiro apresenta 48 casas. Sendo 44 de ATIVIDADES divididas em quatro grupos:

- Atividades executadas pelo Gerente - Roxo
- Atividades executadas pelo Analista - Rosa
- Atividades executadas pelo Desenvolvedor - Verde
- Atividades executadas pelo Testador – Laranja

As outras casas são de FERRAMENTAS, SORTE/REVÉS, PAPEL, E $$.

OBJETIVO: Organizar um processo de software de maneira mais completo em relação às etapas de um ciclo de vida e mais completo em relação à norma ISO/IEC 29110.

JOGADORES: Podem jogar de 2 até 4 equipes de no máximo 4 pessoas cada. A equipe pode escolher a cor de seu peão, colocando-o no ponto de partida. Em seguida embaralham-se as cartas de Sorte e Revés, que devem ser colocadas de cabeça para baixo no centro do tabuleiro.

DINHEIRO: Cada grupo deverá receber o valor de R$2100 divididos em notas de:

- 2 notas de $500,00
- 5 notas de $100,00
- 12 notas de $50,00

QUEM COMEÇA O JOGO: Joga se um dado, e o grupo com o maior número inicia, seguido pelo que tirou o segundo maior e, assim, sucessivamente.

COMEÇO DO JOGO: A primeira equipe lança os dados e, conforme o número de pontos obtidos, avança o seu peão no sentido horário para o espaço atingido. Em um só espaço podem parar vários peões ao mesmo tempo.

Se cair em uma tarefa é dada a opção de comprá-la ou não, caso queira compra-la é necessário responder uma pergunta feita pela outra equipe, em caso de resposta correta a equipe adquire a carta e deverá anotá-lo em seu cartão de processos.

Tirando uma dupla (2 e 2, 3 e 3, etc.) a equipe tem direito a novo lançamento.

PAPÉIS: Antes de o jogo começar, os grupos deverão definir quem interpretará cada papel no grupo. Na primeira rodada as equipes poderão comprar 2 papéis. Os papéis funcionam como critério de capacidade para poder avançar no jogo, uma vez que algumas ferramentas só poderão ser manipuladas por alguém que tenha um determinado papel e de acordo com seu nível (Júnior, Sênior). A equipe só poderá comprar uma carta de papel Sênior se já possuir a de Júnior. A carta de Gerente só possui apenas um nível (equivalente ao Sênior). Caso uma equipe tenha menos pessoas que os papéis previstos no jogo, alguns membros da equipe deverão acumular papéis.

O grupo poderá escolher se quer promover um dos papéis toda vez que passar pela casa de papéis (casa inicial do jogo). Para promover, é preciso pagar um valor especificado na carta de papel. Cada papel possui uma bonificação diferente e dá acesso a mais ferramentas, podendo assim acumular mais pontos. O responsável, por representar a equipe no tabuleiro, será o que estiver com o papel de Gerente, caso a equipe não tenha comprado este papel, um membro será escolhido pela equipe para representá-la. Os papéis são:

Papel	**Tipo 1**	**Valor 1**	**Tipo 2**	**Valor 2**
Gerente	-	-	-	$300,00
Analista	Júnior	$100,00	Sênior	$300,00
Desenvolvedor	Júnior	$100,00	Sênior	$300,00
Testador	Júnior	$100,00	Sênior	$300,00

FERRAMENTAS: As ferramentas são cartas que ajudam a realizar as tarefas que se encontram no tabuleiro. Elas são de diferentes tipos (Gerenciamento, Análise e Projeto, Codificação e Testes). Elas podem ser adquiridas ao longo do jogo para auxiliar no processo. Algumas ferramentas só podem ser utilizadas por determinados papéis (Gerente, Analista, Desenvolvedor ou Testador) e níveis (Júnior, Sênior ou Uso Geral). Elas variam de preço de acordo com a sua qualidade/posição no mercado/etc. Você poderá adquirir ferramentas sempre que passar pela casa FERRAMENTAS no tabuleiro. Os valores são:

Tipo	**Valor**
Uso Geral	$100,00
Júnior	$150,00
Sênior	$200,00

ATIVIDADES: Toda vez que o peão parar em um espaço de ATIVIDADE você poderá comprá-la pelo preço de $100. Mas para isso deverá responder a uma PERGUNTA corretamente, senão o valor subirá para $200. Ao comprar, você receberá o título daquela atividade. Se não possuir a FERRAMENTA adequada para a ATIVIDADE será cobrado $50,00 a mais. Ao término do jogo, se não possuir a carta de FERRAMENTA correspondente à ATIVIDADE, sua pontuação será reduzida à metade. Caso tenha a carta de PAPEL sênior correspondente à ATIVIDADE ganhará 50 pontos por ATIVIDADE correspondente.

As equipes deverão colocar as ATIVIDADES corretamente ao tipo em que ela pertence. Não o fazendo, cada equipe perderá 50 pontos por ATIVIDADE. Os tipos são:

- Planejamento
- Execução e Monitoração
- Análise e Projeto
- Codificação/Testes/Entrega

Obs.: As ATIVIDADES não precisam ser colocadas na ordem dentro do tipo ao que ela pertence.

PERGUNTAS: Antes de poder comprar uma carta de um determinado tipo, é necessário responder uma pergunta sobre Engenharia de Software. Se a equipe souber responder, paga o valor de face. Não sabendo responder, paga o dobro pela atividade. As perguntas podem ter respostas diferentes do gabarito desde que as equipes participantes concordem que a resposta dada também é válida.

SORTE/REVÉS: Você deverá tirar SORTE/REVÉS sempre que parar na casa de mesmo nome. Tire uma carta e execute a ordem imediatamente. Depois devolva a carta para baixo do baralho. Toda vez que um peão adversário parar em uma ATIVIDADE que outra equipe possui, ele deverá retirar a carta de SORTE/REVÉS. Caso dê sorte o bônus será revertido para quem possui a ATIVIDADE, caso dê REVÉS a penalidade será para a equipe que parou na ATIVIDADE de outra equipe (nesse caso o dinheiro será dado ao banco).

PAGAMENTOS: Os pagamentos devem ser efetuados sempre em dinheiro. Se o grupo não tiver dinheiro para pagar, pode-se desfazer de suas propriedades obedecendo a seguinte ordem de negociações:
- Devolver ferramentas.
- Diminuir o nível de algum papel.
- Devolver atividades.

Cada vez que o grupo alcançar a casa demarcada por $$ receberá ($200,00) como bonificação, exceto na primeira rodada.

FALÊNCIA: Se mesmo após vender as ferramentas, devolver as atividades ou diminuir o nível do papel não conseguir pagar suas dívidas ele irá à falência, e se retirará do jogo.
OBS.: Durante um jogo nenhum jogador poderá dar ou emprestar dinheiro a outro.

TÉRMINO DO JOGO: O jogo termina após 60 minutos (ou outra duração acordada entre as equipes). Quem conseguir encaixar o maior número de atividades coerentes ao processo (baseados na ISO/IEC 29110) nas fases corretas, utilizando da melhor forma os papéis e as ferramentas para conseguir uma maior bonificação, será declarado como vencedor. Para isso, deve-se:
- Para cada atividade para qual a equipe tenha adquirido uma ferramenta para executá-la, deve-se contar o valor da atividade de face ($100).
- Para cada atividade para qual a equipe não tenha adquirido uma ferramenta para executá-la, deve-se contar o valor da atividade de face ($50).
- Se a equipe tiver algum papel Sênior, ganha-se uma bonificação de $50,00 para as atividades que podem ser por aquele papel.
- Se a equipe tiver alguma ferramenta Sênior, ganha-se uma bonificação de $50,00 para as atividades para a qual aquela ferramenta é necessária.
- Para cada atividade associada a uma fase incorreta, há uma penalidade de $50,00.

Boa sorte às equipes!

LISTA DE PERGUNTAS SOBRE O CONTEÚDO DO JOGO

#	Pergunta	Resposta
1	Quais destes modelos de ciclo de vida é considerado ágil: RUP, Espiral, XP?	XP
2	É possível combinar modelos de ciclo de vida?	Sim. Os modelos iterativos, incrementais e ágeis podem ser combinados. Uma vez que o projeto de software pode ser flexível, a análise deverá ser feita em conjunto com o cliente e definir se os requisitos permitem esta combinação.
3	Processo unificado e UML são a mesma coisa?	O Processo Unificado é a metodologia de desenvolvimento de software orientado a objetos que usa a UML, que por sua vez é a metodologia que permite a modelagem orientada a objetos de sistema.
4	É possível provar que um componente de software e até mesmo um programa inteiro está correto. Por que isto não é feito com frequência?	Porque não é economicamente viável.
5	Dentre os 3 princípios de processos ágeis, cite aquele que está errado.	.Acolha bem os pedidos de alterações, mesmo atrasados no desenvolvimento. Os processos ágeis se aproveitam das mudanças como uma vantagem competitiva na relação com o cliente. . Atenção contínua para com a excelência técnica e para com bons projetos aumenta a agilidade . A equipe se avalia para ver como tornar-se mais eficiente ao final de todo o projeto.
6	Para que servem as reuniões diárias de um projeto SCRUM?	A reunião Scrum, realizada diariamente, ajuda a equipe a revelar problemas potenciais o mais cedo possível. Ela também leva á "socialização do conhecimento" e, portanto, promove uma estrutura de equipe auto-organizada.
7	Cite algumas razões para as frequentes mudanças de requisito em um projeto	A complexidade dos sistemas impõe mudanças à medida que se adquire maior conhecimento sobre os mesmos, • Requisitos errados ou mal definidos precisam ser corrigidos/ajustados ao longo do processo de desenvolvimento, • Mudanças no ambiente: regras de negócios, leis, políticas internas, • Desenvolvedores querem adicionar funcionalidades mais avançadas de modo a oferecer vantangem • Tecnologia muda, • Clientes mudam de idéia.
8	O que é uma solução pontual na XP?	É a criação imediata de um protótipo operacional quando um problema de projeto é encontrado. O protótipo é testado e avaliado.

#	Pergunta	Resposta
9	É possível ser ágil e ainda sim manter o foco em qualidade? Por quê?	Sim. Porque no desenvolvimento ágil na engenharia de software é necessário definir processos que sejam manipuláveis, adaptáveis, sem excessos. Ainda mais utilizando se de uma filosofia que tem quatro elementos chaves: a importância das equipes se auto-organizarem, que tenham controle sobre o trabalho por ela realizado, sobre a comunicação e sobre a colaboração entre os membros da equipe e entre os desenvolvedores e seus clientes; o reconhecimento de que as mudanças representam oportunidades e ênfase na entrega rápida do software para satisfazer o cliente.
10	Em que consiste um teste bem sucedido?	É aquele que consegue capturar muitos defeitos antes do usuário final.
11	Por que o feedback é importante em uma equipe de sotware?	Porque estabelece os mecanismos de responsabilidades(revisões técnicas por exemplo) para as equipes, e definem uma série de abordagens para correções quando um membro falhar em suas atribuições pois assim se estabelece técnicas baseadas nas equipes voltadas para realimentação(o feedback em si) e resolução de problemas.
12	O que é o modelo de análise?	O modelo de análise preenche a lacuna entre uma descrição sistêmica que descreve o sistema como um todo ou a funcionalidade de negócio que é atingida aplicando-se software, hardware, dados, pessoal e outros elementos de sistema e um projeto de software que descreve a arquitetura, a interface do usuário e a estrutura em termos de componentes de software.
13	O que são modelos de maturidade de software? Cite 2.	Os modelos de maturidade são um metamodelo de processo. Eles surgiram para avaliar a qualidade dos processos de software aplicados em uma organização (empresa ou instituição). CMMI e MPS-BR
14	O que são fluxos de exceções nos casos de uso?	Descrevem os procedimentos que devem ser adotados nos casos de erros durante os fluxos principal e alternativos.
15	É uma boa prática codificar logo depois de um modelo de análise ter sido criado?	Não. Falta modelo de projeto (que indica a solução tecnológica).
16	Qual o propósito da análise de domínio?	É a identificação, a análise e a especificação de requisitos comuns de um campo de aplicação específico, tipicamente para reutilização em vários projetos dentro deste campo de aplicação.
17	O que é um pacote de análise?	É uma parte da modelagem de análise, essa parte é a categorização, vários elementos do modelo de análise(casos de uso, classes de análise) são categorizado de modo que são empacotados em um agrupamento que recebe um nome representativo, a isso chamamos pacote de análise.

#	Pergunta	Resposta
18	O que é a matriz de rastreabilidade?	Segundo o Guia PMBOK®, a matriz de rastreabilidade dos requisitos liga os requisitos às suas origens e os rastreia durante todo o ciclo de vida do projeto. Seu uso ajuda a garantir que cada requisito adiciona valor de negócio através da sua ligação aos objetivos de negócio e aos objetivos do projeto, além de fornecer um meio de rastreamento do início ao fim do ciclo de vida do projeto, ajudando a garantir que os requisitos aprovados na documentação sejam entregues no final do projeto. Finalmente, fornece uma estrutura de gerenciamento das mudanças do escopo do produto.
19	O que torna o projeto de software diferente da codificação?	o Projeto de Software, o qual traduz, num conjunto de representações gráficas, tabulares ou textuais, os requisitos do software definidos na fase anterior; estas representações (diversas técnicas de representação podem ser adotadas em um mesmo projeto) permitirão definir, com um alto grau de abstração, aspectos do software como a arquitetura, os dados, algoritmos e características da interface; • a Codificação, onde as representações realizadas na etapa de projeto serão mapeadas em uma ou em várias linguagens de programação, a qual será caracterizada por um conjunto de instruções executáveis no computador; nesta etapa, considera-se também a geração de código de implementação, aquele obtido a partir do uso de ferramentas (compiladores, etc...) e que será executado pelo hardware do sistema.
20	Como avaliar a qualidade de um projeto de software?	Uma equipe de Garantia da Qualidade elabora um planejamento, supervisão, registros, e relatos de garantias de qualidade, através disso a equipe responsável pelo projeto de software faz o acompanhamento através de checklists de apoio para verificar se tudo está conforme. Características como: Funcionalidade, Confiabilidade, Usabilidade, Eficiência, Manutenibilidade, Portabilidade, Efetividade, Segurança, Produtividade e Satisfação.
21	O que é um projeto de software?	De acordo com Pressman, Projeto é a representação significativa de alguma coisa que será construída. Em engenharia de software, o Projeto de Software é a fase de desenvolvimento, na qual são feitos modelos com todas as entidades que serão construídas posteriormente a partir dos requisitos do sistema. O projeto de software foca em 4 áreas, como: dados, arquitetura, interface e componentes. Para garantir que um projeto está sendo feito com qualidade é necessário avaliar continuamente pontos referentes a corretude, completude, clareza e consistência com os requisitos do sistema.

#	Pergunta	Resposta
22	Por que o controle de versão é útil ao projeto?	Um sistema de controle de versão (ou versionamento), na função prática da Ciência da Computação e da Engenharia de Software, é um software com a finalidade de gerenciar diferentes versões no desenvolvimento de um documento qualquer. Esses sistemas são comumente utilizados no desenvolvimento de software para controlar as diferentes versões — histórico e desenvolvimento — dos códigos-fontes e também da documentação.
23	Por que a documentação é importante no projeto?	Porque a documentação é usada para propósitos de futuras manutenções e aprimoramentos, pois auxilia na redução de horas preciosas na correção de problemas ao indicar bug's e pontos vulneráveis no sistema e suas correções. Além disso s houver mudanças na equipe a documentação poderá auxiliar aos novos membros como está a situação do software.
24	O que é um template de documento?	É um documento de conteúdo, com apenas a apresentação visual(apenas cabeçalhos por exemplo)definida pela empresa e instruções sobre onde e qual tipo de conteúdo deve entrar a cada local da apresentação — por exemplo conteúdos que podem aparecer no início e conteúdos que só podem aparecer no final.
25	O que são riscos de um projeto?	São os que ameaçam o plano do projeto. Se os riscos do projeto se tornarem reais, é possível que o cronograma fique atrasado e os custos aumentam. Os riscos de projeto identificam problemas potenciais de orçamento, cronograma, pessoal, recursos, clientes e requisitos e seu impacto sobre o projeto de software
26	O que são pontos de função?	é uma unidade de medida de software para estimar o tamanho de um sistema de informação baseando-se na funcionalidade percebida pelo usuário do sistema, independentemente da tecnologia usada para implementá-lo. Pontos de Função medem o tamanho funcional do software.
27	Por que não se deve aguardar até que todos os testes terminem para descobrir e corrigir todos os erros de software?	Quanto mais cedo se acha um erro, mais barata sua correção. Redução do tempo de identificação e correção dos defeitos. Documentação dos limites do sistema. Evita que falhas detectadas pelos testes se manifestem em versões futuras do programa.
28	O que são testes de caixa branca?	O analista tem acesso ao código fonte, conhece a estrutura interna do produto sendo analisado e possibilita que sejam escolhidas partes específicas de um componente para serem avaliadas. Esse tipo de teste, também conhecido como teste estrutural, é projetado em função da estrutura do componente e permite uma averiguação mais precisa do comportamento dessa estrutura. Perceba que o acesso ao código facilita o isolamento de uma função ou ação, o que ajuda na análise comportamental das mesmas.

#	Pergunta	Resposta
29	O que são teste de caixa preta?	Teste de caixa-preta é um teste de software para verificar a saída dos dados usando entradas de vários tipos. Tais entradas não são escolhidas conforme a estrutura do programa.
30	O que é um teste unitário?	Também conhecida como teste unitário ou teste de módulo, é a fase em que se testam as menores unidades de software desenvolvidas (pequenas partes ou unidades do sistema).O universo alvo desse tipo de teste são as sub-rotinas, métodos, classes ou mesmo pequenos trechos de código. Assim, o objetivo é o de encontrar falhas de funcionamento dentro de uma pequena parte do sistema funcionando independentemente do todo.
31	Um programa pode ser correto e ainda assim não apresentar boa qualidade? Explique.	Sim. O fato de ele estar correto apenas indica que suas funcionalidades estão operacionais. Mas apesar disto, nada garante que ele foi feito com boas práticas de Engenharia de Software. Sem testes, documentação ou não atendendo todos os requisitos dos clientes.
32	O que é o teste de integração?	Teste de integração é a fase do teste de software em que módulos são combinados e testados em grupo. Ela sucede o teste de unidade, em que os módulos são testados individualmente, e antecede o teste de sistema, em que o sistema completo (integrado) é testado num ambiente que simula o ambiente de produção.
33	Qual a diferença entre verificação e validação?	Verificação - Avaliar se o que foi planejado realmente foi realizado. Ou seja, se os requisitos e funcionalidades documentados foram implementados, além disso, a verificação também pode ser realizada para especificação de sistemas, para avaliar se os requisitos estão sendo documentados como deveriam e ainda prever falhas ou inconsistências entre requisitos. Validação - Avaliar se o que foi entregue atende as expectativas do cliente. Ou seja, se os requisitos estão sendo implementados para atender a regra de negócio do cliente. A validação final do sistema é realizada pelo próprio cliente.
34	Por que um modelo altamente acoplado é difícil de testar em unidade?	O acoplamento é uma medida de interconexão entre módulos de um sistema, depende da complexidade da interface entre módulos. ele é medido entre dois módulos, definindo o grau de dependência entre eles. Quanto mais dependente for um do outro mais difícil será testar em unidade.

#	Pergunta	Resposta
35	O que é o KanBan?	Kanban é um termo de origem japonesa e significa literalmente “cartão” ou “sinalização”. É um conceito relacionado com a utilização de cartões (post-it e outros) para indicar o andamento dos fluxos de produção em empresas de fabricação em série. Nesses cartões são colocadas indicações sobre uma determinada tarefa, por exemplo, “para executar”, “em andamento” ou “finalizado”. Atualmente, o Kanban é muitas vezes usado em conjunto com o Scrum, porque são duas metodologias usadas no desenvolvimento ágil de software.
36	Qual destas afirmações não é uma vantagem sobre o modelo de ciclo de vida incremental?	Entregas parciais facilitam a identificação e correção de erros entre os componentes do software; Facilita a manutenção dos “módulos”; O fim do processo pode ser previamente definido.
37	Cite 2 características de modelo de ciclo de vida em espiral.	Por ser incremental podem ser adicionadas novas funcionalidades em cada nova versão; Praticamente não existe distinção entre desenvolvimento e pós-entrega; Maior controle sobre os riscos do projeto, tornando o processo de construção de um produto complexo mais seguro; Modelo destina-se exclusivamente a desenvolvimento de software interno; A abordagem deste modelo exige grande experiência na avaliação dos riscos; O modelo espiral é mais adequado para sistemas complexos e que exijam um alto nível de interações com os usuários desse sistema.
38	Qual destas afirmações não é uma vantagem sobre o modelo de ciclo de vida iterativo?	A cada ciclo do sistema os usuários e cliente poderão utilizar o sistema diretamente, eles são os "testadores" no processo de desenvolvimento e eles estarão interagindo com o sistema durante o desenvolvimento; Alterações nos requisitos podem ser rapidamente incorporadas no processo de desenvolvimento; Durante o processo de desenvolvimento não há a necessidade de adaptar e refinar o sistema.
39	Cite 2 características do modelo de ciclo de vida cascata.	Torna o processo de desenvolvimento estruturado; Tem uma ordem sequencial de fases; Cada fase cai em cascata na próxima e cada fase deve estar terminada antes do início da seguinte; Todas as atividades identificadas nas fases do modelo são fundamentais e estão na ordem certa.

#	Pergunta	Resposta
40	O que é uma baseline?	É um conceito de gerenciamento de configuração de software que nos ajuda a controlar as mudanças, sem impedir seriamente as mudanças justificáveis. No contexto de engenharia de software, definimos uma baseline como um marco de referência no desenvolvimento de um software, que é caracterizado pela entrega de um ou mais itens de configuração e pela aprovação desses, obtida por meio de uma revisão técnica formal. Ao aprovarmos uma determinada configuração, seja de hardware ou de software, estaremos criando uma baseline.
41	Como é feita uma Revisão por Pares?	A verificação do artefato é realizada por outros desenvolvedores, que não sabem quem foi o autor do artefato, com isso evita-se que o rigor crítico seja baixo e evita atritos.
42	O que são requisitos funcionais?	Um requisito funcional define uma função de um software ou parte dele. Ele é o conjunto de entradas, seu comportamento e sua saída, ou seja, envolve cálculos, lógicas de trabalho, manipulação e processamento de dados, entre outros.
43	O que são requisitos não funcionais?	Requisitos não funcionais são relacionados ao uso da aplicação em termos de desempenho, usabilidade, confiabilidade, disponibilidade, segurança e tecnologias envolvidas. Muitas vezes, os requisitos não funcionais acabam gerando restrições aos funcionais.
44	Acrescentar pessoas a um projeto em atraso necessariamente irá retardá-lo. Verdadeiro ou Falso?	Verdadeiro. O que ocorre é que, quando novas pessoas entram, as que já estavam terão de gastar tempo situando os recém-chegados, reduzindo, consequentemente, o tempo destinado ao desenvolvimento produtivo. Podem-se adicionar pessoas, mas somente de forma planejada e bem coordenada.
45	Como funciona um cronograma de projeto se software?	É uma atividade que distribui o esforço estimado por toda a duração planejada do projeto alocando esse esforço para tarefas específicas de engenharia de software. Conforme o projeto caminha, cada item é refinado em um cronograma detalhado. Nesse momento, ações e tarefas de software específicas são identificadas e dispostas em um cronograma.
46	Para que serve o diagrama de Gantt?	O Gráfico de Gantt, ou Diagrama de Gantt, é uma ferramenta que permite a visualização do progresso de projetos. É, basicamente, uma ferramenta de acompanhamento, bastante visual, cujo objetivo é deixar a informação o mais transparente possível. A função do Gráfico de Gantt é de uma forma genérica, permitir o acompanhamento da realização de tarefas e atividades que compõem um projeto de objetivo maior. Dessa forma, a utilização da ferramenta permite que todos os interessados possam verificar o andamento das atividades.

#	Pergunta	Resposta
47	O que é o caminho crítico de um projeto?	O caminho crítico é a sequência de atividades que devem ser concluídas nas datas programadas para que o projeto possa ser concluído dentro do prazo final. Se o prazo final for excedido, é porque no mínimo uma das atividades do caminho crítico não foi concluída na data programada.
48	O que são riscos conhecidos e riscos previsíveis?	riscos conhecidos - são aqueles que podem ser descobertos após uma cuidadosa avaliação do plano do projeto, do ambiente comercial e técnico no qual o projeto está sendo desenvolvido e de outras fontes de informação confiáveis.(por exemplo, data de entrega irreal, falta de documentação dos requisitos ou do escopo do software, ambiente de desenvolvimento ruim). riscos previsíveis - são oriundos da experiência d projetos anteriores(por exemplo, rotatividade do pessoal, comunicação deficiente com o cliente, diluição do esforço da equipe conforme as solicitações de manutenção vão sendo atendidas).
49	O que é mitigação de riscos?	Na Gerência de Projetos de Software, o plano de mitigação inclui procedimentos para amenizar ou eliminar a ocorrência dos riscos impactantes no projeto.
50	O que é o CMMI?	O CMMI foi baseado nas melhores práticas para desenvolvimento e manutenção de produtos. Há uma ênfase tanto em engenharia de sistemas quanto em engenharia de software, e há uma integração necessária para o desenvolvimento e a manutenção.
51	O que é o MPS-BR?	O MPS.BR ou Melhoria de Processos do Software Brasileiro é simultaneamente um movimento para a melhoria da qualidade (Programa MPS.BR) e um modelo de qualidade de processo (Modelo MPS). Voltado para a realidade do mercado de pequenas e médias empresas de desenvolvimento de software no Brasil, ele é baseado nas normas ISO/IEC 12207 e ISO/IEC 15504 e compatível com o CMMI. Uma das principais vantagens do modelo é seu custo reduzido de certificação em relação as normas estrangeiras, sendo ideal para micro, pequenas e médias empresas que são a grande maioria no Brasil.
52	O que é qualidade de processo de software?	A qualidade no processo procura identificar a má qualidade o quanto antes, o que é feito pelo controle da conformidade à especificação, e corrigir o problema, evitando que continue o desperdício até o fim. Para garantir a conformidade à especificação ao longo do processo, é necessário especificar como executar atividades e seus resultados e controlar sistematicamente todo esse processo que irá atingir a qualidade. É preciso ainda identificar e eliminar as fontes da má qualidade, mediante alterações apropriadas no processo, ou seja, nas especificações de suas atividades.

#	Pergunta	Resposta
53	O que é qualidade de produto de software?	é a rigorosa definição das características relevantes do produto, estabelecendo os atributos e as variáveis que deve conter cuja dimensão deve ser assegurada. A especificação é o documento que formalizará essas definições.
54	O que é um software "bom o suficiente"?	São versões iniciais do software colocadas no mercado por decisões estratégicas. Ele fornece funções e características de alta qualidade que os usuários desejam, mas ao mesmo tempo, fornece outras funções e características mais obscuras ou especializadas e ainda contendo erros conhecidos.
55	O que é gerência de configuração?	É a área da Engenharia de Software responsável por fornecer apoio para o desenvolvimento do software. Tem o propósito de estabelecer e manter integridade de todos os produtos do trabalho de um processo ou projeto e disponibilizá-lo a todos os envolvidos.
56	O que você entende por terceirizar?	Contratação de serviços que poderiam ser fornecidos pela organização se esta tivesse capacidade ou vontade de exercer estas funções. Pode ser mais econômico contratar serviços do que mantê-los dentro da organização.
57	Qual a diferença entre o SCRUM e o XP?	O foco do SRUM se concentra mais na parte gerencial, por sua vez o XP foca mais nas áreas de engenharia e programação.
58	O que é engenharia de software?	é a aplicação de uma abordagem sistemática, disciplinada e quantificável no desenvolvimento, na operação e na manutenção de software, na operação e na manutenção do software.
59	O que é processo de software?	No contexto da engenharia de software, é definido como uma metodologia para as atividades, ações e tarefas necessárias para desenvolver um software de alta qualidade.
60	Verdadeiro ou Falso: A engenharia de software nos fará criar documentação volumosa e desnecessária e invariavelmente irá nos retardar?	A engenharia de software não trata da criação de documentos, trata da criação de um produto de qualidade. Melhor qualidade conduz à redução do retrabalho, e menos retrabalho resulta em maior rapidez na entrega. A necessidade de documentação deve ser adaptada a cada contexto de desenvolvimento e a cada empresa.

FORMULÁRIO DE PERGUNTAS SOBRE O JOGO

Feedback sobre o Jogo ProcSoft

Ao responder esse questionário, você autoriza que as informações sejam utilizadas de forma anônima para a avaliação do jogo ProcSoft. Não há respostas certas ou erradas.

O formulário é dividido em duas partes. As perguntas 1 a 9 estão relacionados à sua opinião sobre o jogo. As perguntas 10 a 15 estão relacionados ao quanto jogo consegui transmitir conhecimentos relacionados a processos de software em geral e à norma ISO/IEC 29910, em particular.

Obrigado.

1) Você gostou de jogar o ProcSoft?	( ) Sim	( ) Não	( ) Indiferente
2) Você achou a jogabilidade complicada?	( ) Sim	( ) Não	( ) Indiferente
3) Você diria que aprendeu com o jogo?	( ) Sim	( ) Não	( ) Indiferente
4) Entre o jogo e uma aula, você prefere:	( ) Aula	( ) Jogo	( ) Aula e Jogo
5) Uma versão digital do jogo atrairia mais a sua atenção?	( ) Sim	( ) Não	( ) Indiferente
6) Você jogaria o ProcSoft de novo?	( ) Sim	( ) Não	( ) Indiferente

7) O que mais você gostou no jogo?

8) O que menos você gostou no jogo?

9) Você tem alguma sugestão de melhoria para o jogo?

10) Como uma empresa de software pode utilizar a norma ISO/IEC 29110?

11) Quais os elementos mínimos são necessários para se definir um Processo de Software?

12) Que atividades, em geral, ficam a cargo do:

- Gerente de Projeto?
- Analista?
- Desenvolvedor?
- Testador?

13) Você acha que a estrutura do processo formado ao longo do jogo é adequada para uma empresa pequena? Por quê?

14) Você acha que as perguntas de sorte/revés são representativas para uma empresa de software? Por quê?

15) Você acha que as perguntas de Engenharia de Software estão de alguma forma ligadas às atividades que o processo apresenta? Comente se achar necessário.

ANEXO III

Tabuleiro e Gabarito de Respostas

Planejamento do Projeto		Execução e Monitoração		Análise e Projeto		Codificação / Testes / Entrega	

	Papel	Revisar o Escopo do Projeto e definir com cliente as instruções de cada entrega	Corrigir os defeitos encontrados nos testes de software	Definir as tarefas para os membros do equipe de acordo com seus papéis	Formalizar a completude do projeto e realizar entrega de acordo com o Plano do Projeto	Corrigir os defeitos encontrados até que o sucesso dos testes unitários sejam alcançados	Realizar testes de software usando Casos e Procedimentos de testes para integração e produção do software e documentar os resultados no "Test Report"
	Realizar testes de regressão até o critério de saída ser atingido						
	Identificar mudanças nos requisitos e/ou no Plano de Projeto para analisar potenciais riscos e problemas para a conclusão do Projeto						
	Avaliar o progresso do Projeto de acordo com o Plano do Projeto, gerar Relatório de Progresso e tomar ações pertinentes para os problemas e riscos identificados						
	Integrar o software usando componentes de software e atualizar Casos e Procedimentos de testes para integração quando necessário						
	Documentar a versão inicial do Manual do usuário ou atualizar.						

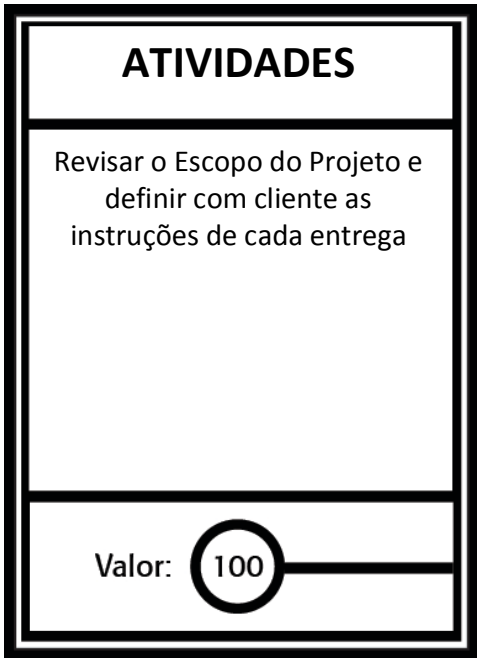
	Planejar o uso de recursos materiais e de hardware e de software e a estratégia de documentação e versionamento						
	Documentar o guia de operação do Produto ou atualizar o existente						
	Documentar ou atualizar as Especificações de Requisitos.						
	Definir o cronograma estimado de cada tarefa do Projeto						
	Incorporar o documento de Especificação de Requisitos e o Manual do usuário na configuração do software na baseline						
	Sorte / **Revés**	Atualizar o repositório do Projeto	Verificar e obter a aprovação do documento de Especificação de Requisitos	Identificar as tarefas e Estabelecer o tempo de duração para cada tarefa	Documentar ou atualizar o Design de Software. Analisar a Especificação de requisitos para gerar a arquitetura do software. Gerar o documento de Rastreabilidade	Construir ou atualizar os componentes de Software baseado no detalhamento do Design de Software	Incorporar o documento de Design de Software e o de Rastreabilidade as configurações de Software como parte da baseline.

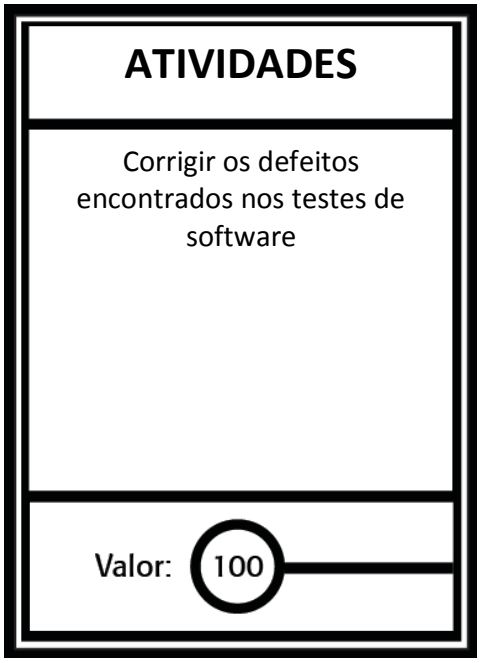
		Validar se a especificação de Requisitos satisfaz e atende as expectativas. Os resultados encontrados são documentados em Validação de Resultados	Identificar os Riscos do Projeto	Criar ou atualizar Casos de Teste e Procedimentos de Testes para testes integrados baseados na Especificação de Requisitos e Design de Software	Entender o Design de Software	Incorporar Casos e procedimentos de Testes, Test Report, Guia de Operação do produto e Documentação do Usuário de Software para as configurações de software como parte da baseline
	Definir a composição da equipe, definindo papéis e responsabilidades					
	Verificar e obter a aprovação do Design de Software. Os resultados serão documentados no documento de Verificação de resultados e se precisar de mudanças começar um pedido de mudanças					
		Analisar e Estimar Pedidos de Mudança				
		Conduzir Reuniões com Equipe e com o Cliente				

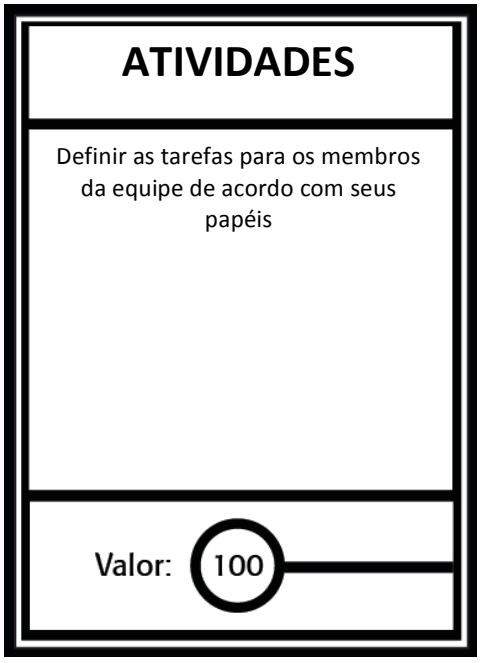
	Ferramenta	Calcular o custo/esforço do Projeto	Fazer backup de acordo com a Estratégia de Controle de Versão	Montar/ Atualizar o ambiente de Implementação	Criar o documento de Manutenção ou atualizar o existente	Aprovar plano do projeto com os interessados
	Colocar casos de testes e procedimentos de teste no Repositório do Projeto					
	Incorporar os componentes de software e a rastreabilidade nas configurações de software como parte da baseline					
	Documentar a Documentação do Usuário do Software ou atualizar a existente.					
	Criar ou atualizar os casos de teste de unidade e aplicá-los para ver se os componentes de Software implementam o que foi detalhado no design de software					

Anexo III

CARTAS DE ATIVIDADES







ATIVIDADES

Corrigir os defeitos encontrados até que o sucesso dos testes unitários seja alcançado

Valor: 100

ATIVIDADES

Realizar testes de software usando Casos e Procedimentos de testes para integração e produção do software e documentar os resultados no "Test Report"

Valor: 100

ATIVIDADES

Definir a composição da equipe, definindo papéis e responsabilidades

Valor: 100

ATIVIDADES

Verificar e obter a aprovação do Design de Software. Os resultados serão documentados no documento de Verificação de resultados e se precisar de mudanças começar um pedido de mudanças

Valor: 100

ATIVIDADES

Analisar e Estimar Pedidos de Mudança

Valor: 100

ATIVIDADES

Conduzir Reuniões com Equipe e com o Cliente

Valor: 100

ATIVIDADES

Atualizar o repositório do Projeto

Valor: 100

ATIVIDADES

Verificar e obter a aprovação do documento de Especificação de Requisitos

Valor: 100

ATIVIDADES

Identificar as tarefas e Estabelecer o tempo de duração para cada tarefa

Valor: 100

ATIVIDADES

Documentar ou atualizar o Design de Software. Analisar a Especificação de requisitos para gerar a arquitetura do software. Gerar o documento de Rastreabilidade

Valor: 100

ATIVIDADES

Construir ou atualizar os componentes de Software baseado no detalhamento do Design de Software

Valor: 100

ATIVIDADES

Incorporar o documento de Design de Software e o de Rastreabilidade as configurações de Software como parte da baseline.

Valor: 100

ATIVIDADES

Criar ou atualizar os casos de teste de unidade e aplicá-los para ver se os componentes de Software implementam o que foi detalhado no design de software

Valor: 100

ATIVIDADES

Documentar a Documentação do Usuário do Software ou atualizar a existente.

Valor: 100

ATIVIDADES

Incorporar os componentes de software e a rastreabilidade nas configurações de software como parte da baseline

Valor: 100

ATIVIDADES

Colocar casos de testes e procedimentos de teste no Repositório do Projeto

Valor: 100

ATIVIDADES

Calcular o custo/esforço do Projeto

Valor: 100

ATIVIDADES

Fazer backup de acordo com a Estratégia de Controle de Versão

Valor: 100

ATIVIDADES

Montar/ Atualizar o ambiente de Implementação

Valor: 100

ATIVIDADES

Criar o documento de Manutenção ou atualizar o existente

Valor: 100

ATIVIDADES

Aprovar plano do projeto com os interessados

Valor: 100

ATIVIDADES

Incorporar Casos e procedimentos de Testes, Test Report, Guia de Operação do produto e Documentação do Usuário de Software para as configurações de software como parte da baseline

Valor: 100

ATIVIDADES

Entender o Design de Software

Valor: 100

ATIVIDADES

Criar ou atualizar Casos de Teste e Procedimentos de Testes para testes integrados baseados na Especificação de Requisitos e Design de Software

Valor: 100

ATIVIDADES

Identificar os Riscos do Projeto

Valor: 100

ATIVIDADES

Validar se a especificação de Requisitos satisfaz e atende as expectativas. Os resultados encontrados são documentados em Validação de Resultados

Valor: 100

ATIVIDADES

Realizar testes de regressão até o critério de saída ser atingido

Valor: 100

ATIVIDADES

Identificar mudanças nos requisitos e/ou no Plano de Projeto para analisar potenciais riscos e problemas para a conclusão do Projeto

Valor: 100

ATIVIDADES

Avaliar o progresso do Projeto de acordo com o Plano do Projeto, gerar Relatório de Progresso e tomar ações pertinentes para os problemas e riscos identificados

Valor: 100

ATIVIDADES

Integrar o software usando componentes de software e atualizar Casos e Procedimentos de testes para integração quando necessário

Valor: 100

ATIVIDADES

Documentar a versão inicial do Manual do usuário ou atualizar.

Valor: 100

ATIVIDADES

Planejar o uso de recursos materiais e de hardware e de software e a estratégia de documentação e versionamento

Valor: 100

ATIVIDADES

Documentar o guia de operação do Produto ou atualizar o existente

Valor: 100

ATIVIDADES

Documentar ou atualizar as Especificações de Requisitos.

Valor: 100

ATIVIDADES

Definir o cronograma estimado de cada tarefa do Projeto

Valor: 100

ATIVIDADES

Incorporar o documento de Especificação de Requisitos e o Manual do usuário na configuração do software na baseline

Valor: 100

CARTAS DE PERGUNTAS

PERGUNTA

Qual destes modelos de ciclo de vida é considerado ágil: RUP, Espiral, XP?

RESPOSTA

XP.

PERGUNTA

É possível combinar modelos de ciclo de vida?

RESPOSTA

Sim. Os modelos iterativos, incrementais e ágeis podem ser combinados. Uma vez que o projeto de software pode ser flexível, a análise deverá ser feita em conjunto com o cliente e definir se os requisitos permitem esta combinação.

PERGUNTA

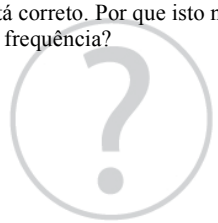
Processo unificado e UML são a mesma coisa?

RESPOSTA

O Processo Unificado é a metodologia de desenvolvimento de software orientado a objetos que usa a UML, que por sua vez é a metodologia que permite a modelagem orientada a objetos de sistema.

PERGUNTA

É possível provar que um componente de software e até mesmo um programa inteiro está correto. Por que isto não é feito com frequência?

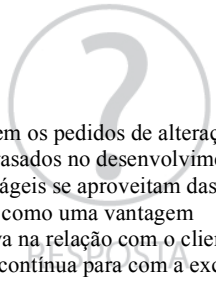


RESPOSTA

Porque não é economicamente viável.

PERGUNTA

Dentre os 3 princípios de processos ágeis, cite aquele que está errado.



.Acolha bem os pedidos de alterações, mesmo atrasados no desenvolvimento. Os processos ágeis se aproveitam das mudanças como uma vantagem competitiva na relação com o cliente;
. Atenção contínua para com a excelência técnica e para com bons projetos aumenta a agilidade;
. A equipe se avalia para ver como tornar-se mais eficiente ao final de todo o projeto.

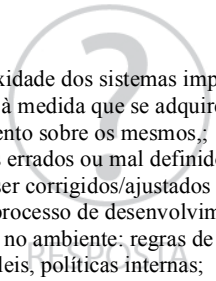
PERGUNTA

Para que servem as reuniões diárias de um projeto SCRUM?

A reunião Scrum, realizada diariamente, ajuda a equipe a revelar problemas potenciais o mais cedo possível. Ela também leva a "socialização do conhecimento" e, portanto, promove uma estrutura de equipe auto-organizada.

PERGUNTA

Cite algumas razões para as frequentes mudanças de requisito em um projeto



A complexidade dos sistemas impõe mudanças à medida que se adquire maior conhecimento sobre os mesmos,;
Requisitos errados ou mal definidos precisam ser corrigidos/ajustados ao longo do processo de desenvolvimento;
Mudanças no ambiente: regras de negócios, leis, políticas internas;
Desenvolvedores querem adicionar funcionalidades mais avançadas de modo a oferecer vantagem;
Tecnologia muda;
Clientes mudam de ideia.

PERGUNTA

O que é uma solução pontual na XP?

É a criação imediata de um protótipo operacional quando um problema de projeto é encontrado. O protótipo é testado e avaliado.

PERGUNTA

É possível ser ágil e ainda sim manter o foco em qualidade? Por quê?

Sim. No desenvolvimento ágil na engenharia de software é necessário definir processos que sejam manipuláveis, adaptáveis, sem excessos. Utilizando se de uma filosofia que tem quatro elementos chaves: a importância das equipes se auto-organizarem, que tenham controle sobre o trabalho por ela realizado, sobre a comunicação e sobre a colaboração entre os membros da equipe e entre os desenvolvedores e seus clientes; o reconhecimento de que as mudanças representam oportunidades e ênfase na entrega rápida do software para satisfazer o cliente.

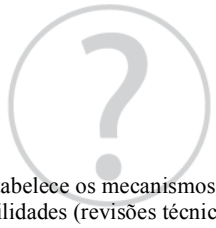
PERGUNTA

Em que consiste um teste bem sucedido?

É aquele que consegue capturar muitos defeitos antes do usuário final.

PERGUNTA

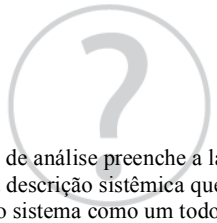
Por que o feedback é importante em uma equipe de software?



Porque estabelece os mecanismos de responsabilidades (revisões técnicas, por exemplo) para as equipes, e definem uma série de abordagens para correções quando um membro falhar em suas atribuições pois assim se estabelece técnicas baseadas nas equipes voltadas para realimentação(o feedback em si) e resolução de problemas.

PERGUNTA

O que é o modelo de análise?



O modelo de análise preenche a lacuna entre uma descrição sistêmica que descreve o sistema como um todo ou a funcionalidade de negócio que é atingida aplicando-se software, hardware, dados, pessoal e outros elementos de sistema e um projeto de software que descreve a arquitetura, a interface do usuário e a estrutura em termos de componentes de software.

PERGUNTA

O que são modelos de maturidade de software? Cite 2.

RESPOSTA

Os modelos de maturidade são um metamodelo de processo. Eles surgiram para avaliar a qualidade dos processos de software aplicados em uma organização (empresa ou instituição). CMMI e MPS-BR.

PERGUNTA

O que são fluxos de exceções nos casos de uso?

RESPOSTA

Descrevem os procedimentos que devem ser adotados nos casos de erros durante os fluxos principal e alternativos.

PERGUNTA

É uma boa prática codificar logo depois de um modelo de análise ter sido criado?

RESPOSTA

Não. Falta modelo de projeto (que indica a solução tecnológica).

PERGUNTA

Qual o propósito da análise de domínio?

RESPOSTA

É a identificação, a análise e a especificação de requisitos comuns de um campo de aplicação específico, tipicamente para reutilização em vários projetos dentro deste campo de aplicação.

PERGUNTA

O que é um pacote de análise?

É uma parte da modelagem de análise, essa parte é a categorização, vários elementos do modelo de análise (casos de uso, classes de análise) são categorizados de modo que são empacotados em um agrupamento que recebe um nome representativo, a isso chamamos pacote de análise.

PERGUNTA

O que é a matriz de rastreabilidade?

Segundo o Guia PMBOK®, a matriz de rastreabilidade dos requisitos liga os requisitos às suas origens e os rastreia durante todo o ciclo de vida do projeto. Seu uso ajuda a garantir que cada requisito adiciona valor de negócio através da sua ligação aos objetivos de negócio e aos objetivos do projeto, além de fornecer um meio de rastreamento do início ao fim do ciclo de vida do projeto, ajudando a garantir que os requisitos aprovados na documentação sejam entregues no final do projeto. Fornece uma estrutura de gerenciamento das mudanças do escopo do produto.

PERGUNTA

O que torna o projeto de software diferente da codificação?

O Projeto de Software traduz num conjunto de representações gráficas, tabulares ou textuais, os requisitos do software definidos na fase anterior; estas representações (permitirão definir, com um alto grau de abstração, aspectos do software como a arquitetura, os dados, algoritmos e características da interface);
A Codificação, onde as representações realizadas na etapa de projeto serão mapeadas em uma ou em várias linguagens de programação, nesta etapa, considera-se também a geração de código de implementação, aquele obtido a partir do uso de ferramentas e que será executado pelo hardware do sistema.

PERGUNTA

Como avaliar a qualidade de um projeto de software?

Uma equipe de Garantia da Qualidade elabora um planejamento, supervisão, registros, e relatos de garantias de qualidade, através disso a equipe responsável pelo projeto de software faz o acompanhamento através de checklists de apoio para verificar se tudo está conforme. Características como: Funcionalidade, Confiabilidade, Usabilidade, Eficiência, Manutenibilidade, Portabilidade, Efetividade, Segurança, Produtividade e Satisfação.

PERGUNTA

O que é um projeto de software?

De acordo com Pressman, Projeto é a representação significativa de alguma coisa que será construída. Em engenharia de software, o Projeto de Software é a fase de desenvolvimento, na qual são feitos modelos com todas as entidades que serão construídas posteriormente a partir dos requisitos do sistema. O projeto de software foca em 4 áreas, como: dados, arquitetura, interface e componentes. Para garantir que um projeto está sendo feito com qualidade é necessário avaliar continuamente pontos referentes a corretude, completude, clareza e consistência com os requisitos do sistema.

PERGUNTA

Por que o controle de versão é útil ao projeto?

Um sistema de controle de versão (ou versionamento), na função prática da Ciência da Computação e da Engenharia de Software, é um software com a finalidade de gerenciar diferentes versões no desenvolvimento de um documento qualquer. Esses sistemas são comumente utilizados no desenvolvimento de software para controlar as diferentes versões — histórico e desenvolvimento — dos códigos-fontes e também da documentação.

PERGUNTA

Por que a documentação é importante no projeto?

Porque a documentação é usada para propósitos de futuras manutenções e aprimoramentos, pois auxilia na redução de horas preciosas na correção de problemas ao indicar bug's e pontos vulneráveis no sistema e suas correções. Além disso, se houver mudanças na equipe a documentação poderá auxiliar aos novos membros como está a situação do software.

PERGUNTA

O que é um template de documento?

É um documento de conteúdo, com apenas a apresentação visual(apenas cabeçalhos por exemplo)definida pela empresa e instruções sobre onde e qual tipo de conteúdo deve entrar a cada local da apresentação — por exemplo conteúdos que podem aparecer no início e conteúdos que só podem aparecer no final.

PERGUNTA

O que são riscos de um projeto?

São os que ameaçam o plano do projeto. Se os riscos do projeto se tornarem reais, é possível que o cronograma fique atrasado e os custos aumentam. Os riscos de projeto identificam problemas potenciais de orçamento, cronograma, pessoal, recursos, clientes e requisitos e seu impacto sobre o projeto de software.

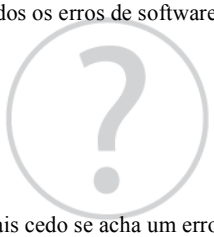
PERGUNTA

O que são pontos de função?

É uma unidade de medida de software para estimar o tamanho de um sistema de informação baseando-se na funcionalidade percebida pelo usuário do sistema, independentemente da tecnologia usada para implementá-lo.
Pontos de Função medem o tamanho funcional do software.

PERGUNTA

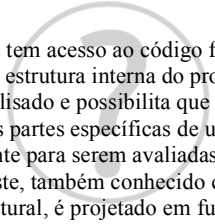
Por que não se deve aguardar até que todos os testes terminem para descobrir e corrigir todos os erros de software?



Quanto mais cedo se acha um erro, mais barata sua correção.
Redução do tempo de identificação e correção dos defeitos.
Documentação dos limites do sistema.
Evita que falhas detectadas pelos testes se manifestem em versões futuras do programa.

PERGUNTA

O que são testes de caixa branca?



O analista tem acesso ao código fonte, conhece a estrutura interna do produto sendo analisado e possibilita que sejam escolhidas partes específicas de um componente para serem avaliadas. Esse tipo de teste, também conhecido como teste estrutural, é projetado em função da estrutura do componente e permite uma averiguação mais precisa do comportamento dessa estrutura. Perceba que o acesso ao código facilita o isolamento de uma função ou ação, o que ajuda na análise comportamental das mesmas.

PERGUNTA

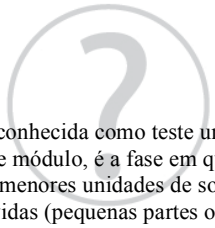
O que são teste de caixa preta?

RESPOSTA

Teste de caixa-preta é um teste de software para verificar a saída dos dados usando entradas de vários tipos. Tais entradas não são escolhidas conforme a estrutura do programa.

PERGUNTA

O que é um teste unitário?



RESPOSTA

Também conhecida como teste unitário ou teste de módulo, é a fase em que se testam as menores unidades de software desenvolvidas (pequenas partes ou unidades do sistema).O universo alvo desse tipo de teste são as sub-rotinas, métodos, classes ou mesmo pequenos trechos de código. Assim, o objetivo é o de encontrar falhas de funcionamento dentro de uma pequena parte do sistema funcionando independentemente do todo.

PERGUNTA

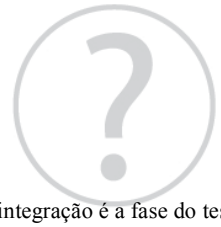
Um programa pode ser correto e ainda assim não apresentar boa qualidade? Explique.

RESPOSTA

Sim. O fato de ele estar correto apenas indica que suas funcionalidades estão operacionais. Mas apesar disto, nada garante que ele foi feito com boas práticas de Engenharia de Software. Sem testes, documentação ou não atendendo todos os requisitos dos clientes.

PERGUNTA

O que é o teste de integração?



RESPOSTA

Teste de integração é a fase do teste de software em que módulos são combinados e testados em grupo. Ela sucede o teste de unidade, em que os módulos são testados individualmente, e antecede o teste de sistema, em que o sistema completo (integrado) é testado num ambiente que simula o ambiente de produção.

PERGUNTA

Qual a diferença entre verificação e validação?

Verificação - Avaliar se o que foi planejado realmente foi realizado. Ou seja, se os requisitos e funcionalidades documentados foram implementados, além disso, a verificação também pode ser realizada para especificação de sistemas, para avaliar se os requisitos estão sendo documentados como deveriam e ainda prever falhas ou inconsistências entre requisitos.

Validação - Avaliar se o que foi entregue atende as expectativas do cliente. Ou seja, se os requisitos estão sendo implementados para atender a regra de negócio do cliente. A validação final do sistema é realizada pelo próprio cliente.

PERGUNTA

Por que um modelo altamente acoplado é difícil de testar em unidade?

O acoplamento é uma medida de interconexão entre módulos de um sistema, depende da complexidade da interface entre módulos. ele é medido entre dois módulos, definindo o grau de dependência entre eles. Quanto mais dependente for um do outro mais difícil será testar em unidade.

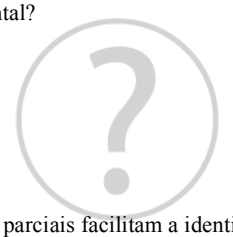
PERGUNTA

O que é o KanBan?

Kanban é um termo de origem japonesa e significa literalmente “cartão” ou “sinalização”. É um conceito relacionado com a utilização de cartões (post-it e outros) para indicar o andamento dos fluxos de produção em empresas de fabricação em série. Nesses cartões são colocadas indicações sobre uma determinada tarefa, por exemplo, “para executar”, “em andamento” ou “finalizado”. Atualmente, o Kanban é muitas vezes usado em conjunto com o Scrum, porque são duas metodologias usadas no desenvolvimento ágil de software.

PERGUNTA

Qual destas afirmações não é uma vantagem sobre o modelo de ciclo de vida incremental?



Entregas parciais facilitam a identificação e correção de erros entre os componentes do software;

Facilita a manutenção dos “módulos”;

O fim do processo pode ser previamente definido.

PERGUNTA

Cite 2 características de modelo de ciclo de vida em espiral.

Por ser incremental podem ser adicionadas novas funcionalidades em cada nova versão;
Praticamente não existe distinção entre desenvolvimento e pós-entrega;
Maior controle sobre os riscos do projeto, tornando o processo de construção de um produto complexo mais seguro;
Modelo destina-se exclusivamente a desenvolvimento de software interno;
A abordagem deste modelo exige grande experiência na avaliação dos riscos;
O modelo espiral é mais adequado para sistemas complexos e que exijam um alto nível de interações com os usuários desse sistema.

PERGUNTA

Qual destas afirmações não é uma vantagem sobre o modelo de ciclo de vida iterativo?

A cada ciclo do sistema os usuários e cliente poderão utilizar o sistema diretamente, eles são os "testadores" no processo de desenvolvimento e eles estarão interagindo com o sistema durante o desenvolvimento;

Alterações nos requisitos podem ser rapidamente incorporadas no processo de desenvolvimento;

Durante o processo de desenvolvimento não há a necessidade de adaptar e refinar o sistema.

PERGUNTA

Cite 2 características do modelo de ciclo de vida cascata.

Torna o processo de desenvolvimento estruturado;
Tem uma ordem sequencial de fases;
Cada fase cai em cascata na próxima e cada fase deve estar terminada antes do início da seguinte;
Todas as atividades identificadas nas fases do modelo são fundamentais e estão na ordem certa.

PERGUNTA

O que é uma baseline?

É um conceito de gerenciamento de configuração de software que nos ajuda a controlar as mudanças, sem impedir seriamente as mudanças justificáveis. No contexto de engenharia de software, definimos uma baseline como um marco de referência no desenvolvimento de um software, que é caracterizado pela entrega de um ou mais itens de configuração e pela aprovação desses, obtida por meio de uma revisão técnica formal. Ao aprovarmos uma determinada configuração, seja de hardware ou de software, estaremos criando uma baseline.

PERGUNTA

Como é feita uma Revisão por Pares?

A verificação do artefato é realizada por outros desenvolvedores, que não sabem quem foi o autor do artefato, com isso evita-se que o rigor crítico seja baixo e evita atritos.

PERGUNTA

O que são requisitos funcionais?

Um requisito funcional define uma função de um software ou parte dele. Ele é o conjunto de entradas, seu comportamento e sua saída, ou seja, envolve cálculos, lógicas de trabalho, manipulação e processamento de dados, entre outros.

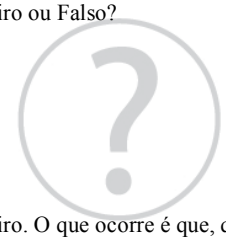
PERGUNTA

O que são requisitos não funcionais?

Requisitos não funcionais são relacionados ao uso da aplicação em termos de desempenho, usabilidade, confiabilidade, disponibilidade, segurança e tecnologias envolvidas. Muitas vezes, os requisitos não funcionais acabam gerando restrições aos funcionais.

PERGUNTA

Acrescentar pessoas a um projeto em atraso necessariamente irá retardá-lo. Verdadeiro ou Falso?



Verdadeiro. O que ocorre é que, quando novas pessoas entram, as que já estavam terão de gastar tempo situando os recém-chegados, reduzindo, consequentemente, o tempo destinado ao desenvolvimento produtivo. Podem-se adicionar pessoas, mas somente de forma planejada e bem coordenada.

PERGUNTA

Como funciona um cronograma de projeto se software?

É uma atividade que distribui o esforço estimado por toda a duração planejada do projeto alocando esse esforço para tarefas específicas de engenharia de software. Conforme o projeto caminha, cada item é refinado em um cronograma detalhado. Nesse momento, ações e tarefas de software específicas são identificadas e dispostas em um cronograma.

PERGUNTA

Para que serve o diagrama de Gantt?

O Gráfico de Gantt, ou Diagrama de Gantt, é uma ferramenta que permite a visualização do progresso de projetos. É, basicamente, uma ferramenta de acompanhamento, bastante visual, cujo objetivo é deixar a informação o mais transparente possível. A função do Gráfico de Gantt é de uma forma genérica, permitir o acompanhamento da realização de tarefas e atividades que compõem um projeto de objetivo maior. Dessa forma, a utilização da ferramenta permite que todos os interessados possam verificar o andamento das atividades.

PERGUNTA

O que é o caminho crítico de um projeto?

O caminho crítico é a sequência de atividades que devem ser concluídas nas datas programadas para que o projeto possa ser concluído dentro do prazo final. Se o prazo final for excedido, é porque no mínimo uma das atividades do caminho crítico não foi concluída na data programada.

PERGUNTA

O que são riscos conhecidos e riscos previsíveis?

Riscos Conhecidos - são aqueles que podem ser descobertos após uma cuidadosa avaliação do plano do projeto, do ambiente comercial e técnico no qual o projeto está sendo desenvolvido e de outras fontes de informação (data de entrega irreal, falta de documentação dos requisitos ou do escopo do software, ambiente de desenvolvimento ruim).
Riscos Previsíveis - são oriundos da experiência de projetos anteriores (rotatividade do pessoal, comunicação deficiente com o cliente, diluição do esforço da equipe conforme as solicitações de manutenção vão sendo atendidas).

PERGUNTA

O que é mitigação de riscos?

RESPOSTA

Na Gerência de Projetos de Software, o plano de mitigação inclui procedimentos para amenizar ou eliminar a ocorrência dos riscos impactantes no projeto.

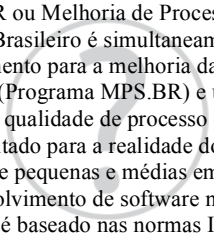
PERGUNTA

O que é o CMMI?

O CMMI foi baseado nas melhores práticas para desenvolvimento e manutenção de produtos. Há uma ênfase tanto em engenharia de sistemas quanto em engenharia de software, e há uma integração necessária para o desenvolvimento e a manutenção.

PERGUNTA

O que é o MPS-BR?



O MPS.BR ou Melhoria de Processos do Software Brasileiro é simultaneamente um movimento para a melhoria da qualidade (Programa MPS.BR) e um modelo de qualidade de processo (Modelo MPS). Voltado para a realidade do mercado de pequenas e médias empresas de desenvolvimento de software no Brasil, ele é baseado nas normas ISO/IEC 12207 e ISO/IEC 15504 e compatível com o CMMI. Uma das principais vantagens do modelo é seu custo reduzido de certificação em relação às normas estrangeiras, sendo ideal para micro, pequenas e médias empresas que são a grande maioria no Brasil.

PERGUNTA

O que é qualidade de processo de software?

A qualidade no processo procura identificar a má qualidade o quanto antes, o que é feito pelo controle da conformidade à especificação, e corrigir o problema, evitando que continue o desperdício até o fim. Para garantir a conformidade à especificação ao longo do processo, é necessário especificar como executar atividades e seus resultados e controlar sistematicamente todo esse processo que irá atingir a qualidade. É preciso ainda identificar e eliminar as fontes da má qualidade, mediante alterações apropriadas no processo, ou seja, nas especificações de suas atividades.

PERGUNTA

O que é qualidade de produto de software?

É a rigorosa definição das características relevantes do produto, estabelecendo os atributos e as variáveis que deve conter cuja dimensão deve ser assegurada. A especificação é o documento que formalizará essas definições.

PERGUNTA

O que é um software "bom o suficiente"?

São versões iniciais do software colocadas no mercado por decisões estratégicas. Ele fornece funções e características de alta qualidade que os usuários desejam, mas ao mesmo tempo, fornece outras funções e características mais obscuras ou especializadas e ainda contendo erros conhecidos.

PERGUNTA

O que é gerência de configuração?

É a área da Engenharia de Software responsável por fornecer apoio para o desenvolvimento do software. Tem o propósito de estabelecer e manter integridade de todos os produtos do trabalho de um processo ou projeto e disponibilizá-lo a todos os envolvidos.

PERGUNTA

O que você entende por terceirizar?

Contratação de serviços que poderiam ser fornecidos pela organização se esta tivesse capacidade ou vontade de exercer estas funções. Pode ser mais econômico contratar serviços do que mantê-los dentro da organização.

PERGUNTA

Qual a diferença entre o SCRUM e o XP?

RESPOSTA

O foco do SRUM se concentra mais na parte gerencial, por sua vez o XP foca mais nas áreas de engenharia e programação.

PERGUNTA

O que é engenharia de software?

RESPOSTA

É a aplicação de uma abordagem sistemática, disciplinada e quantificável no desenvolvimento, na operação e na manutenção de software, na operação e na manutenção do software.

PERGUNTA

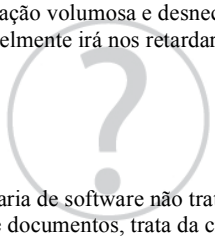
O que é processo de software?

RESPOSTA

No contexto da engenharia de software, é definido como uma metodologia para as atividades, ações e tarefas necessárias para desenvolver um software de alta qualidade.

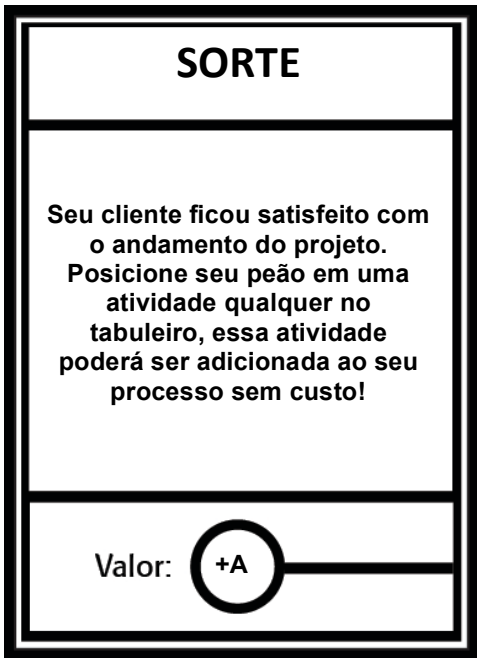
PERGUNTA

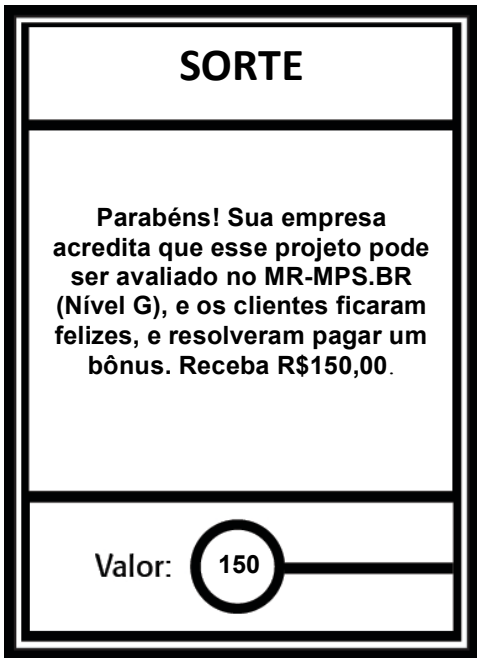
Verdadeiro ou Falso:
A engenharia de software nos fará criar documentação volumosa e desnecessária e invariavelmente irá nos retardar?

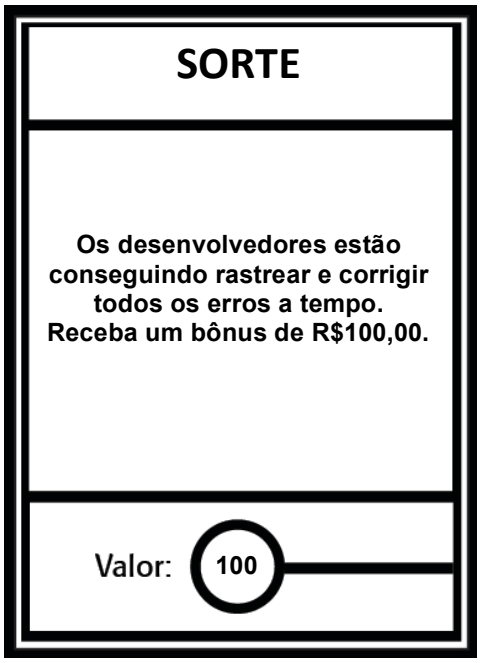


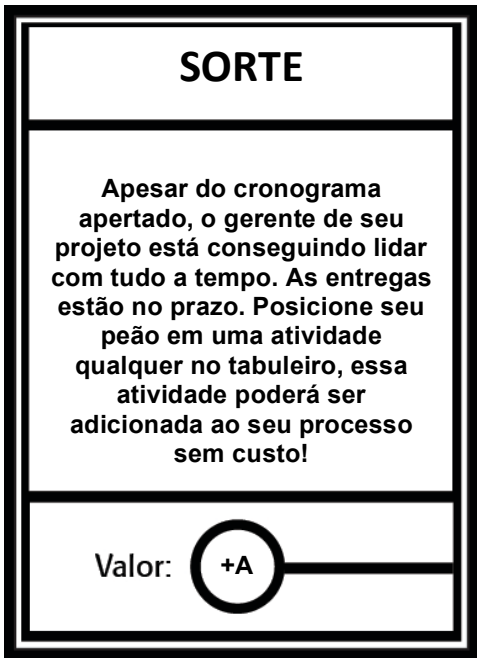
A engenharia de software não trata da criação de documentos, trata da criação de um produto de qualidade. Melhor qualidade conduz à redução do retrabalho, e menos retrabalho resulta em maior rapidez na entrega. A necessidade de documentação deve ser adaptada a cada contexto de desenvolvimento e a cada empresa.

CARTA DE SORTE/REVÉS









SORTE

Foi verificado que sua empresa está com todos seus softwares legalizados, e acabou de se tornar parceiro Ouro de uma fabricante de softwares. Ganhe um bônus de R$150,00.

Valor: 150

SORTE

O controle de versão do seu software e seus repositórios encontram-se atualizados e com backup em dia. Ganhe um bônus de R$50,00

Valor: 50

SORTE

O auditor de qualidade ficou impressionado, pois, no checklist elaborado por ele, suas atividades foram verificadas como conformes. Posicione seu pião em uma atividade qualquer no tabuleiro, essa atividade poderá ser adicionada ao seu processo sem custo!

Valor: +A

SORTE

Com este projeto sua empresa está ganhando mais visibilidade e sendo requisitada por outros clientes. Ganhe um bônus de R$50,00.

Valor: 50

SORTE

As ações de sua empresa subiram na Bolsa de Valores. Ganhe um bônus de R$100,00.

Valor: 100

SORTE

Vocês receberam um adiantamento pelo bom trabalho! Ganhe um bônus de R$100,00.

Valor: 100

REVÉS

Um auditor de qualidade externo viu que seu processo não está de acordo com as normas propostas. Pague uma multa de R$ 100,00.

Valor: -100

REVÉS

Seus templates estão fora do padrão, você terá de padronizá-los! Pague R$50,00.

Valor: -50

REVÉS

O gerente da equipe não planejou o projeto direito e não fez as entregas no prazo. Pague uma multa de R$200,00.

Valor: **-200**

REVÉS

O estagiário da equipe anda muito desleixado, as tarefas não foram cumpridas no prazo. Pague R$50,00.

Valor: **-50**

REVÉS

Houve uma queda de energia e seus arquivos não foram salvos a tempo. Devolva uma atividade.

Valor: **-A**

REVÉS

Descobriram a utilização de software pirata em sua empresa, você terá de se adequar. Pague R$250,00.

Valor: **-250**

REVÉS

O desenvolvedor deixou passar diversos erros no sistema sem corrigi-los. Seu cliente ficou insatisfeito. Pague R$150,00.

Valor: **-150**

REVÉS

A baseline do seu projeto não foi atualizada com as novas modificações, com isso foi entregue uma versão errada do sistema. Devolva uma atividade.

Valor: **-A**

REVÉS

As não conformidades encontradas nos casos de uso pela aplicação do checklist de verificação irão tomar muito tempo para serem corrigidas. Pague R$250,00.

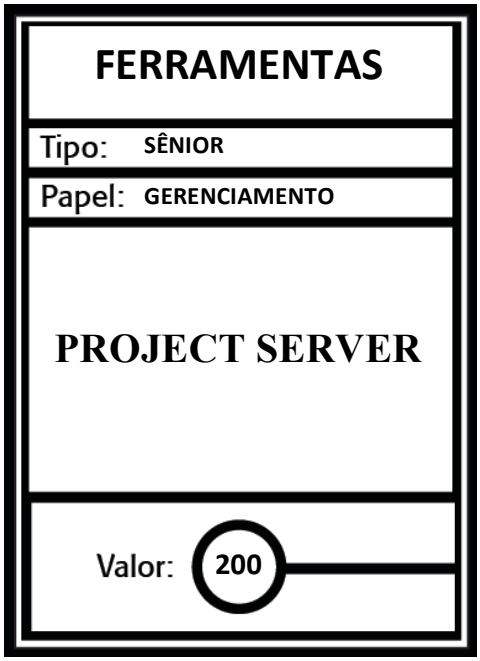
Valor: **-250**

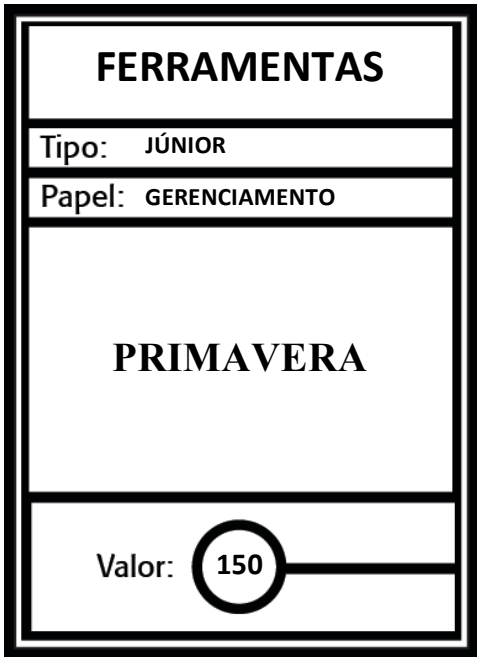
REVÉS

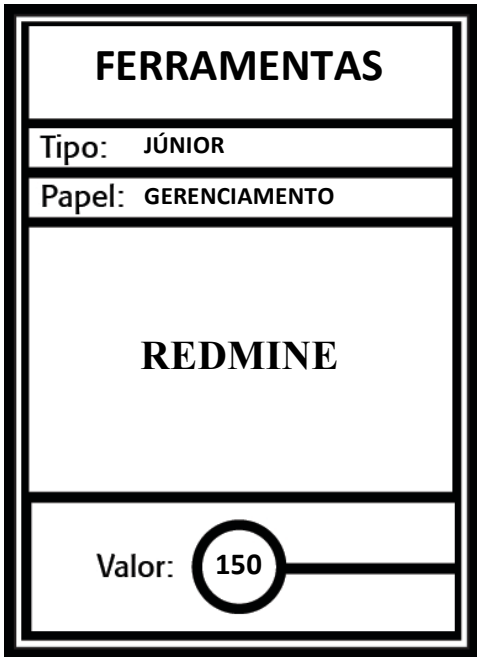
O projeto está atrasado e o gerente da equipe decidiu chamar mais desenvolvedores. Possivelmente o projeto irá atrasar ainda mais. Devolva uma atividade.

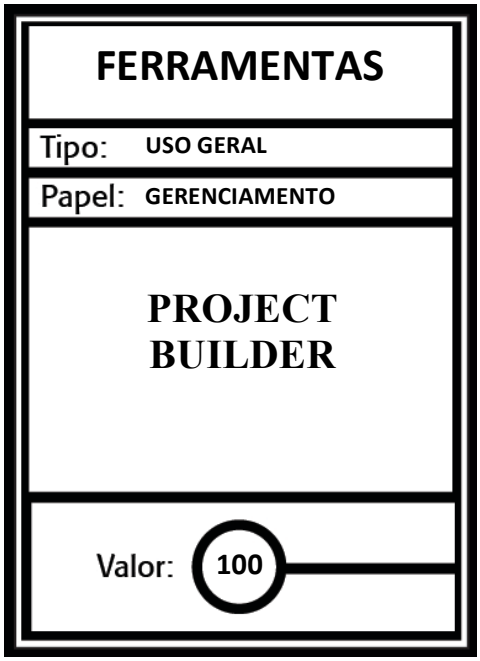
Valor: **-A**

CARTAS DE FERRAMENTAS

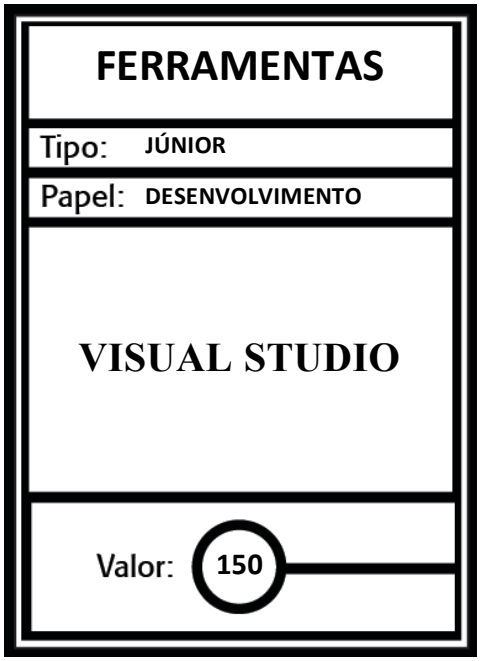


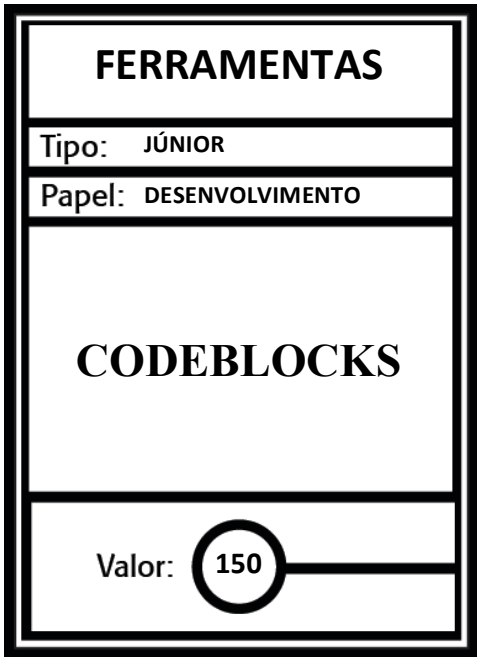


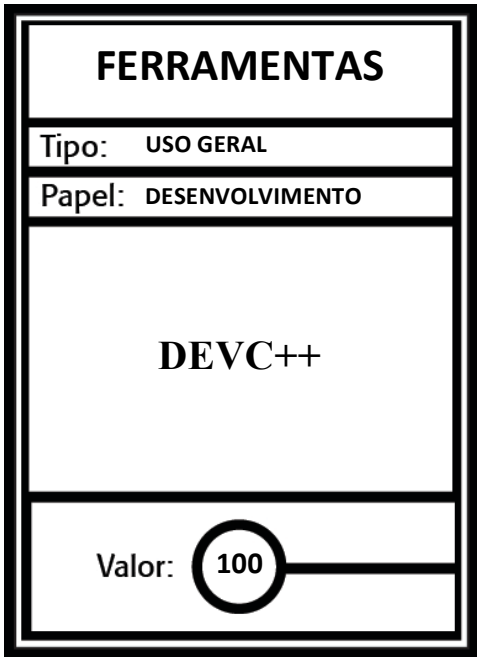


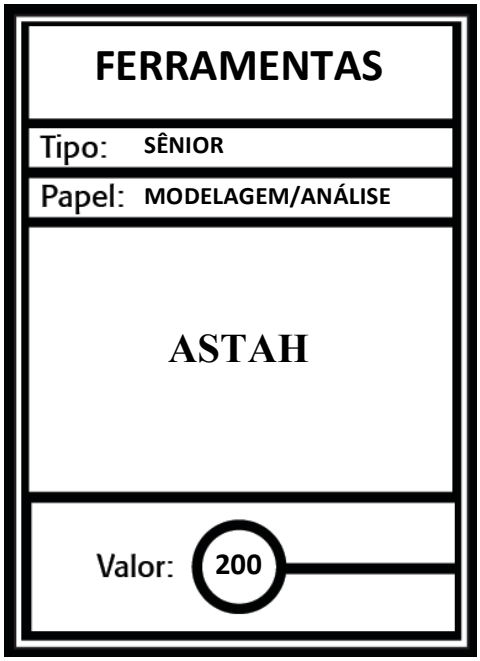


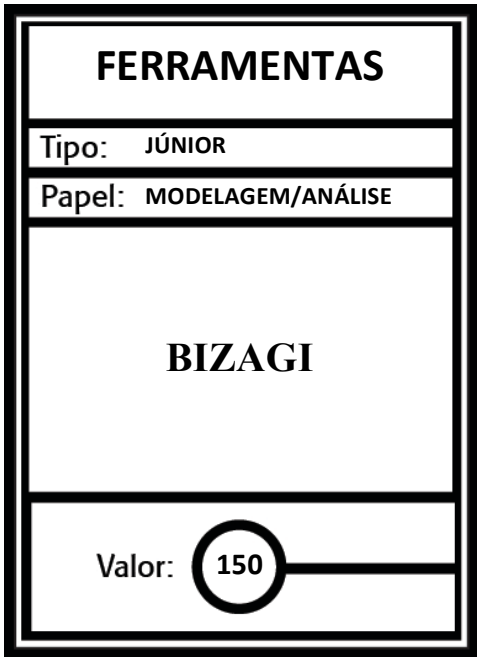

FERRAMENTAS
Tipo: SÊNIOR
Papel: DESENVOLVIMENTO
ECLIPSE
Valor: 200

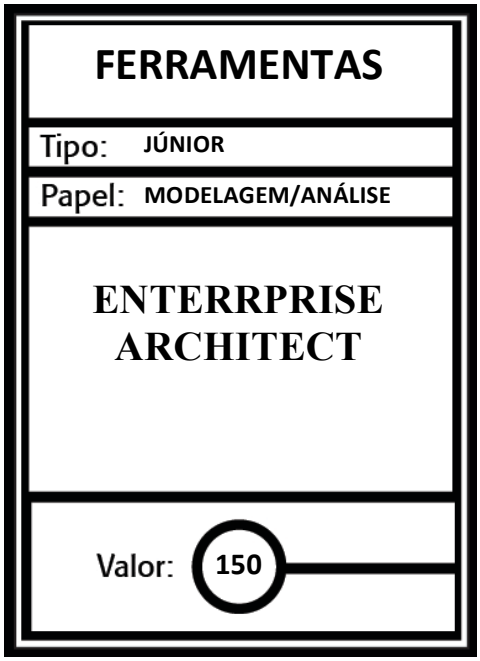

FERRAMENTAS
Tipo: JÚNIOR
Papel: DESENVOLVIMENTO
VISUAL STUDIO
Valor: 150

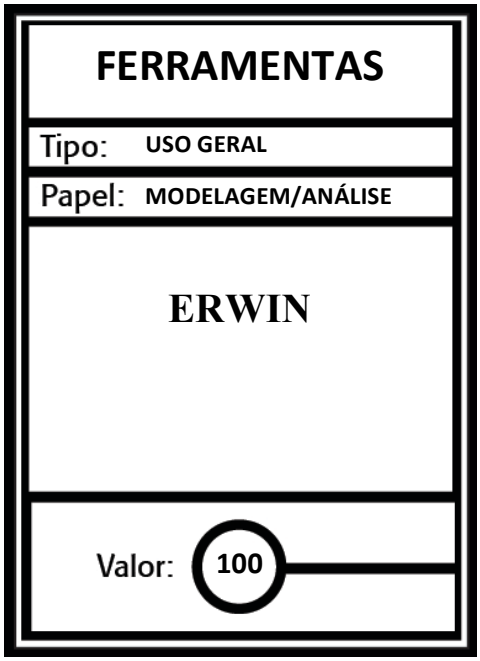

FERRAMENTAS
Tipo: JÚNIOR
Papel: DESENVOLVIMENTO
CODEBLOCKS
Valor: 150

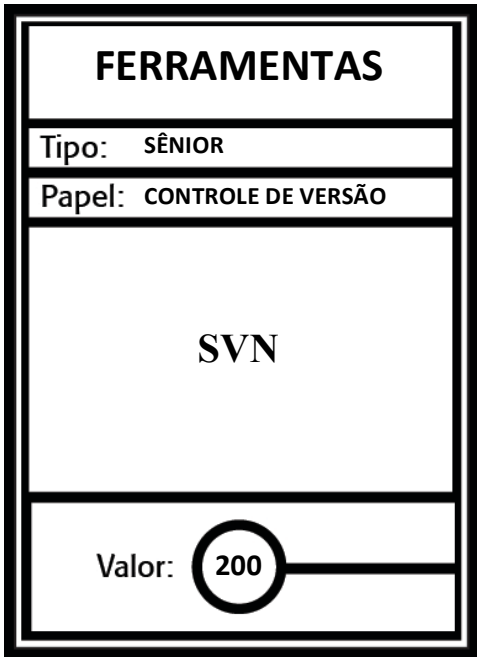

FERRAMENTAS
Tipo: USO GERAL
Papel: DESENVOLVIMENTO
DEVC++
Valor: 100

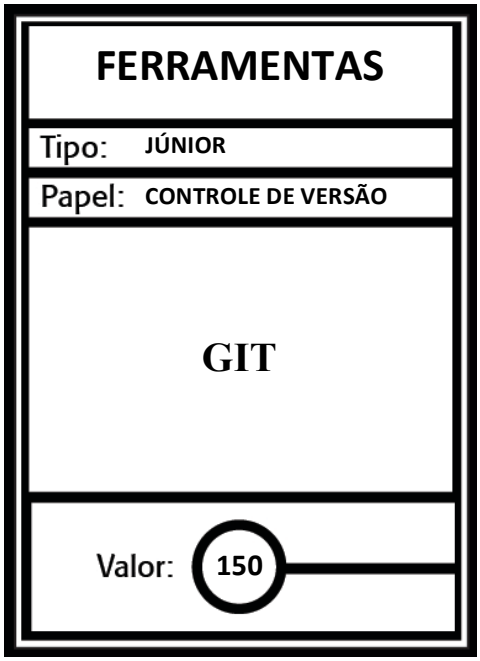

FERRAMENTAS
Tipo: SÊNIOR
Papel: MODELAGEM/ANÁLISE
ASTAH
Valor: 200

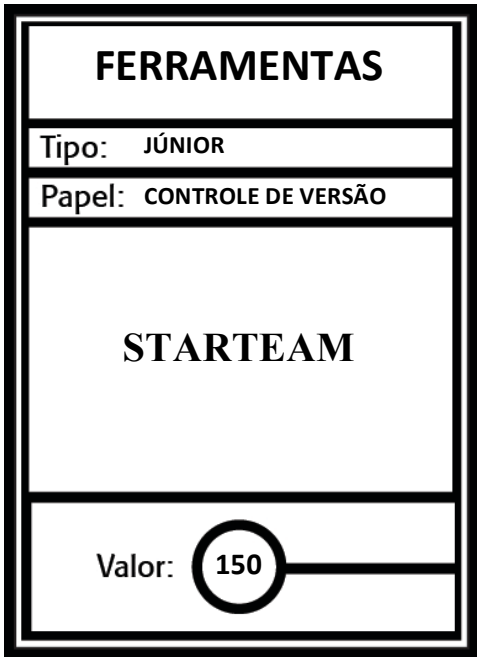

FERRAMENTAS
Tipo: JÚNIOR
Papel: MODELAGEM/ANÁLISE
BIZAGI
Valor: 150


FERRAMENTAS
Tipo: JÚNIOR
Papel: MODELAGEM/ANÁLISE
ENTERRPRISE ARCHITECT
Valor: 150


FERRAMENTAS
Tipo: USO GERAL
Papel: MODELAGEM/ANÁLISE
ERWIN
Valor: 100

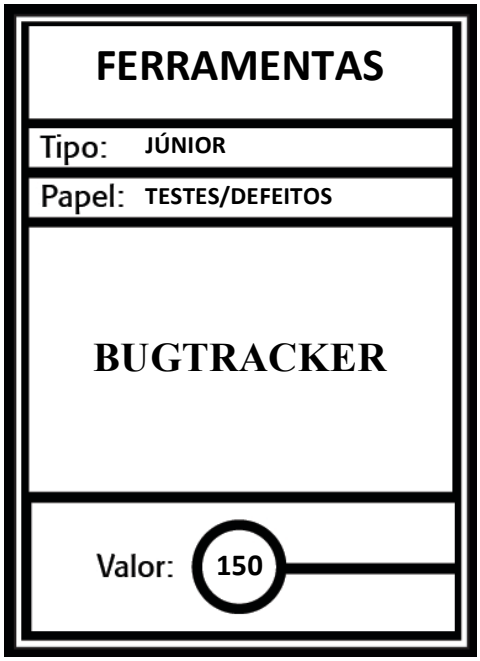

FERRAMENTAS
Tipo: SÊNIOR
Papel: CONTROLE DE VERSÃO
SVN
Valor: 200

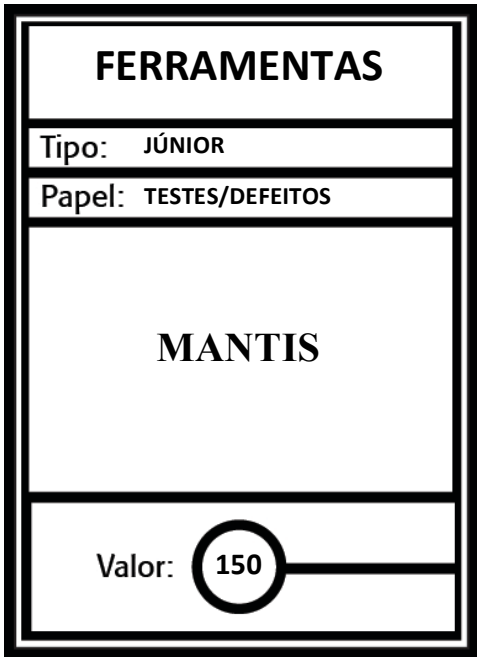

FERRAMENTAS
Tipo: JÚNIOR
Papel: CONTROLE DE VERSÃO
GIT
Valor: 150

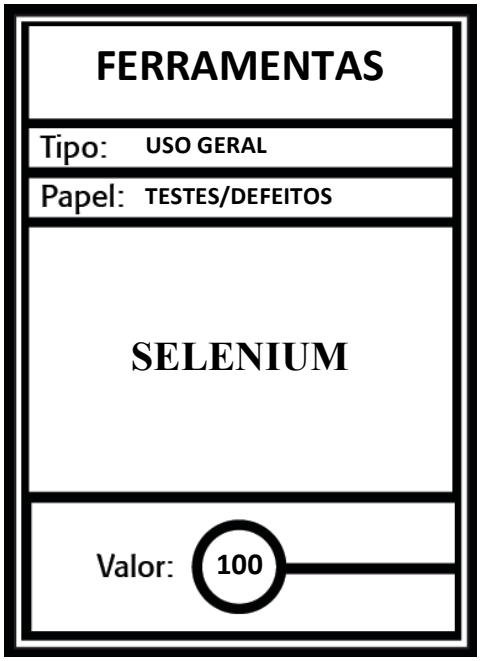

FERRAMENTAS
Tipo: JÚNIOR
Papel: CONTROLE DE VERSÃO
STARTEAM
Valor: 150


FERRAMENTAS
Tipo: USO GERAL
Papel: CONTROLE DE VERSÃO
TFS
Valor: 100


FERRAMENTAS
Tipo: SÊNIOR
Papel: TESTES/DEFEITOS
JUNIT
Valor: 200

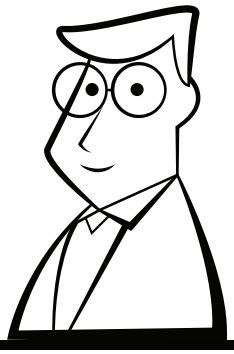

FERRAMENTAS
Tipo: JÚNIOR
Papel: TESTES/DEFEITOS
BUGTRACKER
Valor: 150


FERRAMENTAS
Tipo: JÚNIOR
Papel: TESTES/DEFEITOS
MANTIS
Valor: 150


FERRAMENTAS
Tipo: USO GERAL
Papel: TESTES/DEFEITOS
SELENIUM
Valor: 100

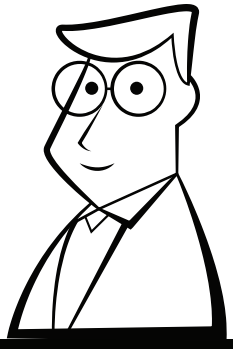
CARTAS DE PAPÉIS

ANALISTA SENIOR



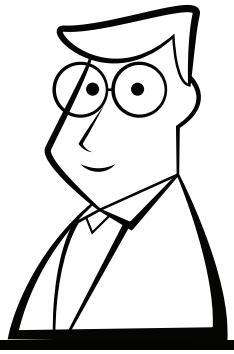
Valor: 300

GERENTE



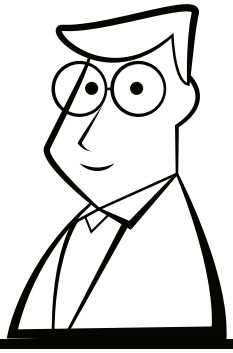
Valor: 300

DESENVOLVEDOR SENIOR



Valor: 300

DESENVOLVEDOR JÚNIOR



Valor: 100

ANALISTA
JÚNIOR

Valor: 100

TESTADOR
SENIOR

Valor: 300

TESTADOR
JÚNIOR

Valor: 100

GABARITO COM AS ATIVIDADES PREVISTAS PARA A DEFINIÇÃO DOS PROCESSOS

Planejamento do Projeto	Execução e Monitoração	
Planejamento do Projeto	**Execução do Plano do Projeto**	**Controle e Avaliação do Projeto**
Revisar o Escopo do Projeto e definir com cliente as instruções de cada entrega	**Analisar e Estimar Pedidos de Mudança**	Avaliar o progresso do Projeto de acordo com o Plano do Projeto, gerar Relatório de Progresso e tomar ações pertinentes para os problemas e riscos identificados
Identificar as tarefas e Estabelecer o tempo de duração para cada tarefa	Conduzir Reuniões com Equipe e com o Cliente	**Identificar mudanças nos requisitos e/ou no Plano de Projeto para analisar potenciais riscos e problemas para a conclusão do Projeto**
Planejar o uso de recursos materiais e de hardware e de software e a estratégia de documentação e versionamento	**Fazer backup de acordo com a Estratégia de Controle de Versão**	
Definir a composição da equipe, definindo papéis e responsabilidades		
Definir o cronograma estimado de cada tarefa do Projeto		
Calcular o custo/esforço do Projeto		
Identificar os Riscos do Projeto		
Aprovar plano do projeto com os interessados		

Análise e Projeto			
Análise dos requisitos de Software	**Arquitetura de Software e Detalhamento do Design**	**Início da Implementação do Software**	**Construção do Software**
Definir as tarefas para os membros do equipe de acordo com seus papéis	**Documentar ou atualizar o Design de Software. Analisar a Especificação de requisitos para gerar a arquitetura do software. Gerar o documento de Rastreabilidade**	Montar/ Atualizar o ambiente de Implementação	Entender o Design de Software
Documentar ou atualizar as Especificações de Requisitos.	*Verificar e obter a aprovação do Design de Software. Os resultados serão documentados no documento de Verificação de resultados e se precisar de mudanças começar um pedido de mudanças*		Construir ou atualizar os componentes de Software baseado no detalhamento do Design de Software
Verificar e obter a aprovação do documento de Especificação de Requisitos	*Criar ou atualizar Casos de Teste e Procedimentos de Testes para testes integrados baseados na Especificação de Requisitos e Design de Software*		*Criar ou atualizar os casos de teste de unidade e aplicá-los para ver se os componentes de Software implementam o que foi detalhado no design de software*
Validar se a especificação de Requisitos satisfaz e atende as expectativas. Os resultados encontrados são documentados em Validação de Resultados	**Incorporar o documento de Design de Software e o de Rastreabilidade as configurações de Software como parte da baseline.**		Corrigir os defeitos encontrados até que o sucesso dos testes unitários sejam alcançados
Documentar a versão inicial do Manual do usuário ou atualizar.	*Colocar casos de testes e procedimentos de teste no Repositório do Projeto*		Incorporar os componentes de software e a rastreabilidade nas configurações de software como parte da baseline
Incorporar o documento de Especificação de Requisitos e o Manual do usuário na configuração do software na baseline			

Codificação/Testes/Entrega		
Integração de Software e Testes	**Fechamento do Projeto**	**Entrega do Produto**
Integrar o software usando componentes de software e atualizar Casos e Procedimentos de testes para integração quando necessário	**Formalizar a completude do projeto e realizar entrega de acordo com o Plano do Projeto**	Criar o documento de Manutenção ou atualizar o existente
Realizar testes de software usando Casos e Procedimentos de testes para integração e produção do software e documentar os resultados no "Test Report"	**Atualizar o repositório do Projeto**	**Realizar entrega de acordo com as Instruções de entrega**
Documentar o guia de operação do Produto ou atualizar o existente		
Documentar a Documentação do Usuário do Software ou atualizar a existente.		
Incorporar Casos e procedimentos de Testes, Test Report, Guia de Operação do produto e Documentação do Usuário de Software para as configurações de software como parte da baseline		
Corrigir os defeitos encontrados nos testes de software		
Realizar testes de regressão até o critério de saída ser atingido		