Controla

Ferramenta de Apoio ao Processo de Desenvolvimento de Software em pequenas empresas

Clayton Vieira Fraga Filho

Com a globalização, os mercados tornaram-se mais competitivos e a demanda por produtos e serviços de melhor qualidade surge como fator decisivo no momento da escolha do parceiro de negócios. Com o desenvolvimento de software não é diferente, o cliente, cada vez mais exigente, demanda por um sistema seguro, que atenda suas necessidades e que seja desenvolvido no menor prazo possível.

As empresas de desenvolvimento de software buscam não somente colocar no mercado produtos e serviços que atendam a demanda de seus clientes, mas também melhorar seus processos de produção. Essa melhoria tem com ponto principal o modelo de produção, levando-se em consideração fatores organizacioanais, ambientais e de pessoal de cada empresa. Assim, toda empresa desenvolvedora de software vive o dilema: Qualidade X Tempo

Este artigo destina-se a descrever a ferramenta **Controla** e suas principais funcionalidades, úteis para pequenas empresas de desenvolvimento de software ou pequenas equipes.

Desenvolvida como trabalho final de Curso Bacharelado em Sistemas de Informação da Faculdade de Viçosa (FDV) – MG, o **Controla** tem como objetivo apoiar as atividades inerentes ao processo de desenvolvimento de software. Por ser gratuita na versão 1.0, pode ser utilizada como alternativa por equipes que optarem por não adquirir uma solução paga para realizarem tal gerenciamento tão importante para o aumento da qualidade em seus produtos.

A definição e implantação de um processo de desenvolvimento de software, sobretudo em pequenas empresas desenvolvedoras de software não é algo comum. Geralmente o foco está no desenvolvimento para o atendimento imediato da necessidade do cliente, ou seja, há pouco ou quase nenhum planejamento antes da codificação. Sendo assim, o gerenciamento dos requisitos e das mudanças ao longo do projeto torna-se um processo bastante dispendioso, sendoo que, na maioria das vezes não há ferramentas de apoio ou documentação formal dos projetos de software, quadro esse agravado pela rotatividade de pessoal (*turnover)*.

O **Controla** oferece importantes recursos. Os mais importantes são relacionados a seguir:

· Gerenciamento de Requisitos;

· Gerenciamento de Casos de Uso;

· Gerenciamento de Casos de Teste e Erros;

· Planejamento de Liberações;

· Gerenciamento de Implementações;

· Controle de Dependência entre implementações;

· Matriz de Rastreabilidade (Traceability Matrix):

o Rastreabilidade dos requisitos;

o Rastreabilidade de projeto;

o Casos de Uso X Implementações;

o Casos de Uso X Casos de Teste;

o Casos de Teste X Erros;

o Implementações X Erros;

o Liberações X Casos de Uso;

o Liberações X Casos de Teste;

o Erros X Liberações;

· Registro de Métricas para todos os artefatos;

· Ferramenta de estimativa de tamanho de software por Pontos de Casos de Uso;

· Ferramenta para priorização de Requisitos;

· Emissão de documentos:

o Documento de Plano de Projeto;

o Documento de Casos de Uso;

o Documento de Especificação de Requisitos;

· Exportação de dados;

Figura 1. Interface principal do Controla

Os *artefatos* são definidos como o conjunto de requisitos, implementações, liberações, casos testes ou erros identificados, produzidos durante o processo de desenvolvimento de software.

Gerenciamento de Requisitos

O **Controla** permite o gerenciamento dos requisitos, ou seja, regras e capacitadas necessárias em um produto de software, utilizando como base descrição breve e completa da necessidade, suas restrições, data de criação, de finalização e responsáveis pela aprovação junto ao cliente.

Outras informações essenciais a um requisito são prioridade, estabilidade e estado, que juntamente com as informações descritivas descrevem completamente o que deve contemplar o software solicitado pelo cliente. Finalmente, um requisito pode ser proposto por mais de um *stakeholder*, sendo assim, na ferramenta é possível associar um ou mais proponentes a um mesmo requisito.

Na definição dos estados dos requisitos de um software, o **Controla** realizada validações, assim a ferramenta restringe operações inválidas por que podem ser executadas pelos analistas durante o gerenciamento destas informações, exibindo mensagens que possam o orientar. Os principais estados identificados para um requisito são:

· Proposto – Indica que o requisito foi solicitado por uma fonte autorizada ;

· Aprovado – Seu impacto no projeto foi analisado, projetado e alocado para uma liberação específica. O usuários concordaram em incorporar o requisito, e o grupo de desenvolvimento de software já aceitou implementar tal requisito.

· Rejeitado – O requisito foi proposto mas não será incorporado em qualquer liberação.

· Em avaliação – O requisito foi proposto e está sendo avaliado por todos os usuários (*stakeholders)* e pela equipe de desenvolvimento.

· Implementado – A implementação do requisito foi projetada, realizada e testada. O requisito foi projeto e associado a um item de implementação.

· Verificado – O requisito foi testado para todo o produto de software, via teste de integração. O requisito foi rastreado aos casos de teste pertinentes. Neste estado o requisito é considerado completo.

· Atendido – O requisito atendeu às expectativas do cliente.

O controle de versão é um importante recurso oferecido nesta ferramenta, pois mantém documentadas as mudanças de cada requisito, possibilitando aos analistas consultarem e confrontarem os dados com a equipe de desenvolvimento e com os clientes, caso necessário. Esta funcionalidade é útil, pois reduz os desgastes provocados por discussões muitas vezes sem fundamento durante o processo de desenvolvimento de software.

Matriz de Rastreabilidade (Traceability Matrix)

No gerenciamento de projetos de *software* as alterações e a avaliação do impacto da mudança durante o ciclo de desenvolvimento é essencial e crítico.

É importante assegurar que, os requisitos possam ser relacionados aos Casos de Uso propostos para o sistema, pois os Casos de Uso esclarecem a implementação proposta do sistema a partir da visão de um usuário. É importante definir um projeto claro para os analistas, portanto se a alteração é proposta enquanto o sistema está em desenvolvimento, o impacto da alteração envolve verificar como a alteração afeta os requisitos, o projeto do sistema e sua implementação.

A matriz de rastreabilidade oferece como grande facilitador a visualização global dos Casos de Uso e requisitos do sistema em uma tabela de forma gráfica, dando suporte ao analista para tomar decisões e descobrir problemas e sua solução de forma mais rápida, muitas vezes antes do fase de implementação. Os artefatos modificados no **Controla** são sinalizados, oferecendo suporte visual ao analista na identificação de potenciais focos de problemas.

Além da matriz de rastreabilidade de projeto, é possível relacionar requisitos, implementação, liberações, testes e erros.

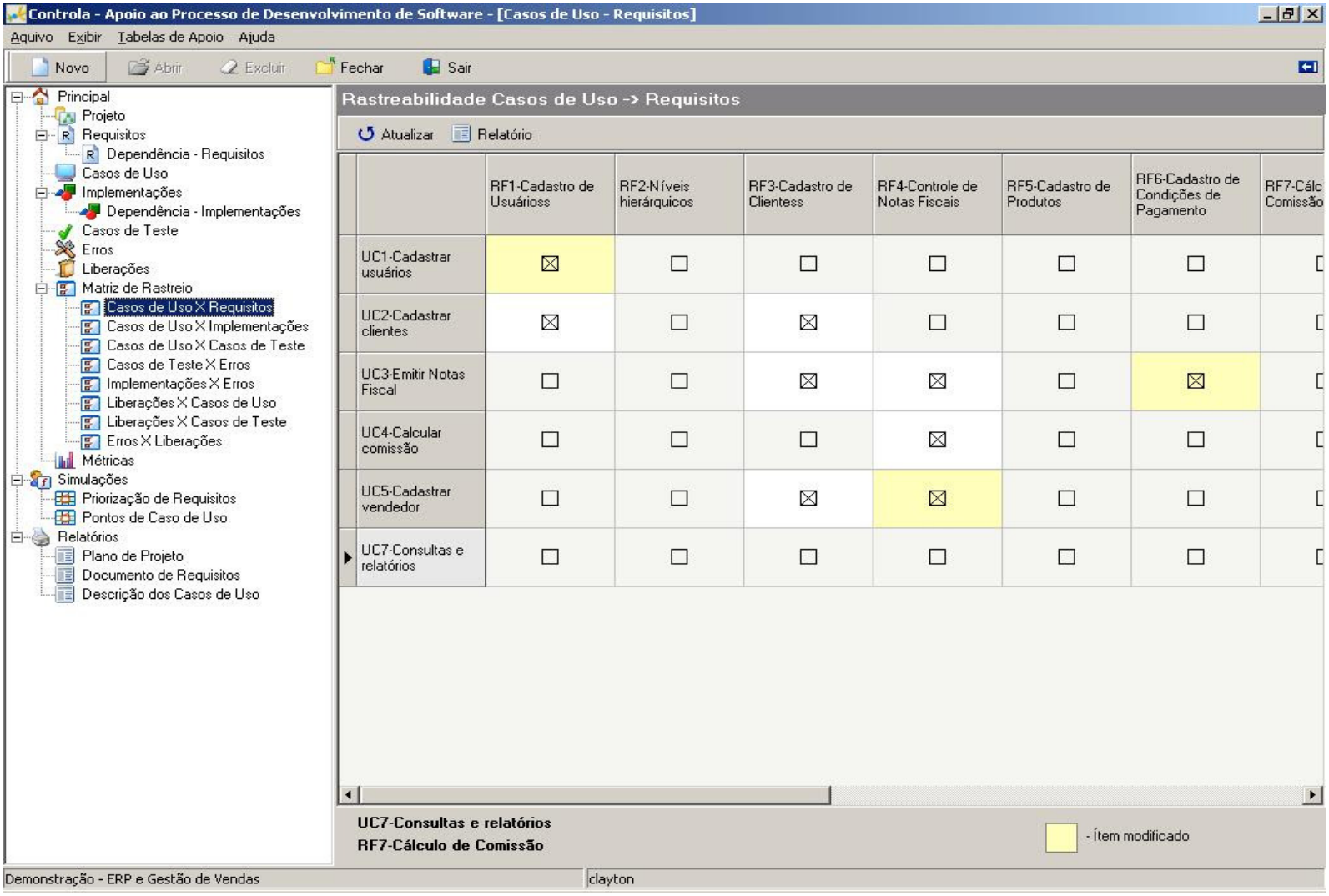

Figura 2. Rastreabilidade de projeto (Casos de Uso X Requisitos).

Durante a operação das matrizes de rastreabilidade, o cadastro de cada artefato de software é atualizado de acordo com a matriz que está sendo atualizada. Como exemplo podemos citar o relacionamento de Casos de Uso com Casos de Teste. O analista pode desejar após ter realizado um relacionammento nesta matriz, que os requisitos e casos de uso associados tenham o estado *Verificado* definido para eles. Neste momento o **Controla** realiza uma validação com o objetivo de consultar se o estado anterior do requisito e casos de uso da relação está definido como *Implementado*, seguindo o fluxo definido, eliminando do analista a responsabilidade de realizar este processo manualmente.

Método de priorização de requisitos baseada em valor, custo e risco

Durante o desenvolvimento de um software, os *stakeholders* podem chegar a um acordo informal quanto à prioridade dos requisitos. Projetos de software exigem uma abordagem mais estruturada que elimine sentimentalismos, políticas e suposições do processo de priorização. Algumas técnicas analíticas e matemáticas envolvem a estimativa do valor e custo relativo de cada requisito (Karl Wiegers 2003). Os requisitos de mais alta prioridade são aqueles que fornecem a maior fração do valor total do produto e a menor fração do custo total (Karlsson and Ryan 1997; Jung 1998 apud Karl Wiegers 2003). Estimar subjetivamente o custo e o valor comparando dois a dois todos os requisitos é impraticável para uma grande quantidade de requisitos.

Outra alternativa é a "*Quality Function Deployment*" (QFD), um método para relacionar o valor que o cliente atribui às características propostas para o produto (Zultner 1993; Cohen 1995). Uma terceira abordagem, importada da Gerência de Qualidade Total (TQM), classifica cada requisito de acordo com vários critérios de sucessos do projeto e calcula uma pontuação para classificar a prioridade dos requisitos. Entretanto, poucas empresas de software parecem estas dispostas a assumir o rigor do QFD e do TQM.

O modelo proposto por Karl Wigers para auxiliar na estimatva das prioridades relativas para um conjunto de requisitos funcionais, adota conceitos do QFD em basear o valor do cliente para o benefício fornecido ao cliente caso um requisito específica seja incorporado, e uma penalidade caso o requisito esteja ausente. A importância de um requisito é diretamente proporcional ao valor que ele agrega e inversamente proporcional ao seu custo e riscos associados à sua implementação. Esta abordagem distribui um conjunto de prioridades estimadas de forma contínua ao invés de simplesmente agrupá-las em um conjunto discreto de poucos níveis de prioridade.

Uma vez que identificados os requisitos essenciais ao produto a ser entregue, o modelo proposto pode ser usado para classificar a importância relativa dos requisitos restantes. Quem deve ser considerado no processo de priorização:

· O gerente do projeto, que conduz o processo, resolve conflitos e ajusta as entradas de outros participantes se necessário;

· Clientes representativos, tais como patrocinadores do produto ou pessoal de marketing, que podem fornecer os valores de benefícios e penalidades;

· Representantes da equipe de desenvolvimento tais como líderes de equipes que podem fornecer os valores de custos e riscos.

Os passos a seguir representam a seqüência para a utilização do modelo de priorização:

1. Os requisitos que devem ser analisados precisam ser incluídos na lista. Todos os ítens devem ser do mesmo nível de abstração – não é permitido comparar requisitos funcionais com Casos de Uso do produto, portanto no **Controla** são incluídos apenas os Requisitos Funcionais. Se certos requisito estão logicamente associadas (por exemplo, você só pode implementar o requisito B somente se o requisito A for incluído), são listados somente requisitos principais (*driving feature*) na análise.

2. O *stakeholder* deve estimar o benefício relativo de cada requisito a ser fornecida para o cliente ou negócios numa escala de 1 a 9. O valor 1 indica que ninguém considera o requisito útil e 9 significa que ele é extramente importante. Estes valores de benefícios indicam um alinhamento dos requisitos com as regras de negócios do produto.

3. Deve ser estimada a penalidade relativa que o cliente ou o negócio sofrerá caso o requisito não seja incluído. Novamente, é necessário utilizar a escala de 1 a 9, onde o valor 1 indica que nada será afetado se o requisito for excluído, e 9 indica um sério problema caso contrário. Requisitos que possuam baixo benefício e baixa penalidade adicionam custo ao projeto, mas agregam pouco valor. Quando os valores de penalidade forem atribuídos, é necessário que se considere o quão infeliz um cliente seria se um requisito específico não fosse incluído. É importante fazer as seguintes perguntas:

· Seu produto perderia pontos em comparação com outro produto que contém tal funcionalidade?

· Haveria alguma conseqüência contratual ou legal?

· Estaria violando algum padrão do governo ou das regras de negócio?

· Os usuários seriam incapazes de executar alguma função necessária?

· Quais problemas surgiriam devido a promessas do pessoal de marketing com relação a uma funcionalide que no final fosse omitida?

4. modelo calcula o valor total de cada requisito como a soma de seus benefícios e penalidades. Por padrão, benefício e penalidades recebem o mesmo peso. Como um refinamento é permitido alterar os pesos relativos para estes dois fatores. .

5. Depois que os desenvolvedores estimarem o custo relativo de implementação de cada requisito, novamente a escala de 1 (fácil e rápido) a 9 (caro e demorado), o modelo calcula o percentual do custo total que cada requisito fornece no modelo. Os desenvolvedores estimam os custos baseados na complexidade, interface, possibilidade de reuso, quantidade de testes, documentação necessária, dentre outros.

6. Similarmente, os desenvolvedores devem estimar o risco relativo de cada requisito na escala de 1 a 9. Requisitos mal-definidos que podem demandar re-trabalho são de alto risco. A planilha calcula o percentual do risco total derivado de cada requisito.

No modelo padrão, o benefício, penalidade, custo e risco são mais pesados, mas no **Controla** é possível ajustar todos os quatro pesos. A precisão desta técnica está limitada à capacidade da equipe em estimar benefícios, penalidades, custos e riscos de cada item.

É importante observar que há pequenas diferenças nas prioridades calculadas. O método não tem formalismo matemático rigoroso. O ideal é agrupar requisitos que tem prioridades similares.

Diferentes usuários freqüentemente têm pensamentos distintos a respeito da importância de um requisito específico ou sobre a penalidade de omití-lo.

É importante tirar os requisitos da teoria para um nível no qual avaliações realistas possam ser feitas. Isto dará ao processo maior chance de construir produtos que agregam o máximo de valor ao negócio.

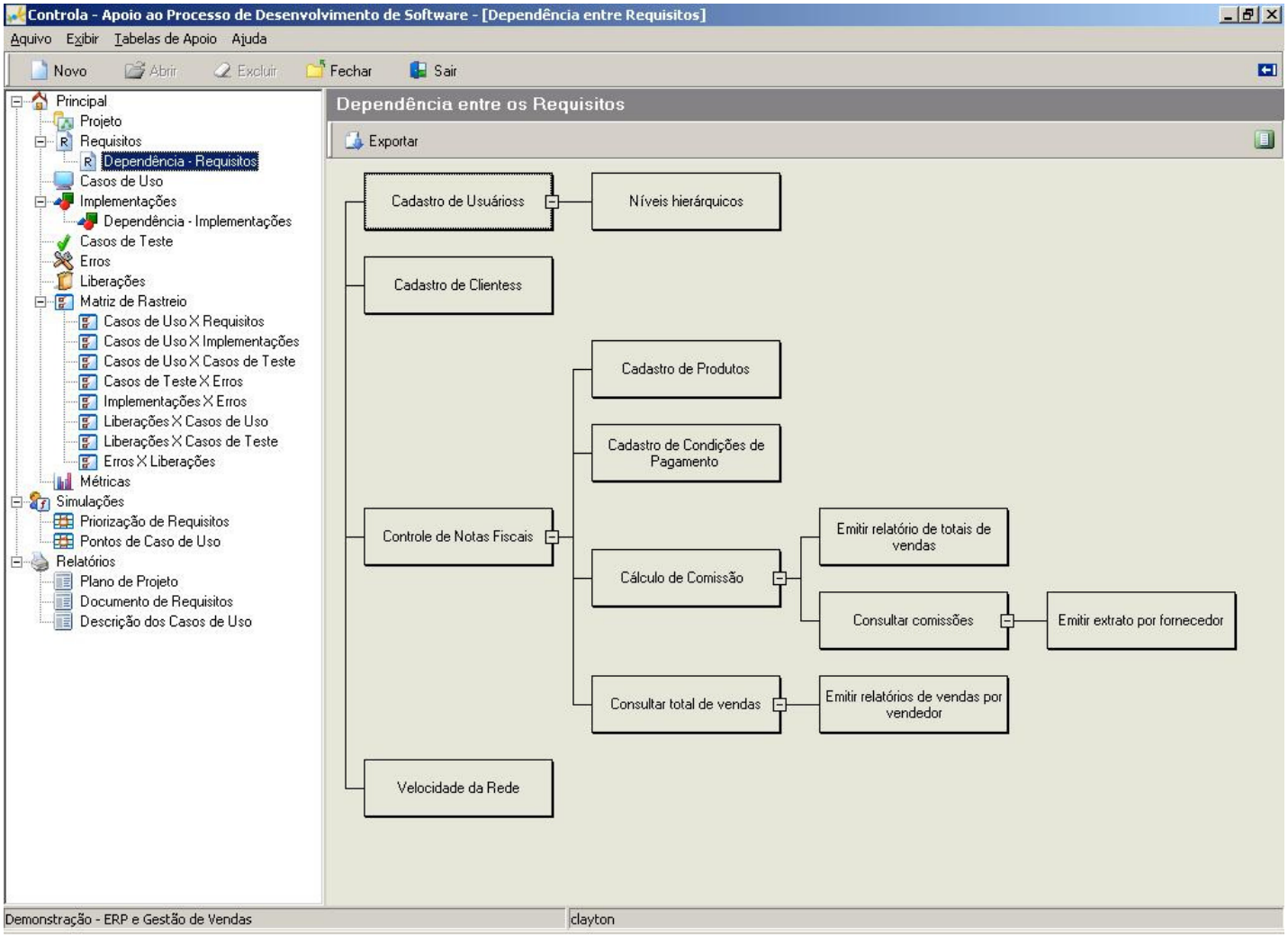

Figura 3. No Controla, é possível criar e exportar diversos cenários utilizando o Método de priorização de requisitos baseada em valor, custo e risco

Conclusões

O **Controla** está sendo utilizado como ferramenta de apoio ao processo de desenvolvimento de software em alguns projetos e incorporando melhorias e ajustes contínuos. A partir deste semestre será utilizado como ferramenta acadêmica na disciplina Engenharia de Software na Faculdade de Viçosa – MG. . A partir da versão 1.0 oferece valor agregado para pequenas equipes que não possuem metodologias definidas e não dispõe de recursos para adquirir as poderosas ferramentas CASE disponíveis no mercado.

Uma versão multi-usuário para ambiente Web está sendo desenvolvida. Possibilitará a colaboração on-line para equipes de desenvolvimento e grupos de *stakeholders* em múltiplos projetos, incluindo recursos de fórum de discussão, workflow e acompanhamento de métricas pré-estabelecidas.

Link para download da ferramenta Controla

http://baixaki.ig.com.br/site/detail34971.htm

Clayton Vieira Fraga Filho (*clayton@cientec.net*), graduado em Sistemas de Informação pela Faculdade de Viçosa (FDV) – MG. É gerente de projetos e desenvolvedor da Cientec Consultoria e Desenvolvimento de Sistemas Ltda.

Alcidelio

Recife – PE, 11 de março de 2013.