Universidade Estadual de Campinas

Faculdade de Tecnologia

Aysy Anne Andrade Duarte
Willian Alves Barboza

Gerenciamento de Projetos Open Source

Limeira, 2010

Universidade Estadual de Campinas

Faculdade de Tecnologia

Aysy Anne Andrade Duarte

Willian Alves Barboza

Gerenciamento de Projetos Open Source

Trabalho apresentado à Faculdade de Tecnologia da UNICAMP.

Limeira, 2010

Resumo

Um desafio constante na produção de software é, além de atender as necessidades dos clientes, é atendê-las respeitando os prazos contratuais com qualidade no produto. Alguns trabalhos acreditam que para chegar ao "*software* ideal" seja necessário seguir regras, metodologias.

Por outro lado surgem projetos *open source* que geralmente não seguem essas regras, mas conseguem chegar ao objetivo final mostrando qualidade e segurança. Para se chegar nessa qualidade, alguns projetos *open source* utilizam a metodologia Bazar, que através da colaboração existente nas comunidades consegue chegar à produção de um software de qualidade, e ainda transfere conhecimento para os demais membros, sem nenhuma remuneração financeira, mas sim pelo reconhecimento pessoal.

Neste trabalho será apresentada breve introdução ao termo open source em seguida uma descrição de como é realizado o gerenciamento de projetos de open source, suas características, regras e aspectos como qualidade.

Abstract

A constant Challenge in software production is not just to see needs customers, but also assist it see the contractual deadline. Some works believe that to reach in "ideal software" is necessary to follow rules and metholologies.

On the other hand appear open source projects that use the methodology Bazaar, which through the existing collaboration in open source communities not only to produce a quality software as well as transfer knowledge to other members without any financial remuneration and over by personal recognition.

Sumário

Lista de Figuras vi
1.1 Motivação 2
1.2 Objetivos 2
1.3 Estrutura do Trabalho 2
Capítulo 2: Processos de Desenvolvimento de Software 3
2.1 Modelos de Processos de Software 3
Capítulo 3: Processo de Desenvolvimento de Software Open Source 6
3.1 Definição 6
3.2 Características dos Projetos Open Source 6
3.3 Organização 7
Capítulo 4. Gerenciamento de Projetos Open Source 10
4.1 Metodologia Bazar 10
4.2 Qualidade 11
4.3 Desenvolvimento 11
4.4 Especificar Requisitos 11
4.5 Lançamento de versões 12
Conclusões 13
Referências Bibliográficas 14

Lista de Figuras

Figura 1. Diagrama do ciclo de vida cascata de Winston Royce na década de 70. 4
Figura 2. Modelo de desenvolvimento evolucionário. 5
Figura 3. Possível diagrama de um projeto *open source*. 7

1. Introdução

Verifica-se um constante desafio na produção de *software,* pois não é essencial apenas atender as necessidades dos clientes, além disso, é necessário atender os prazos, produzir *software* com qualidade e atender as normas contratuais que podem causar alterações no valor final do *software.* Pensando em minimizar imprevistos durante o processo de produção, manutenção e finalização do *software,* surgiram vários estudos acadêmicos que tratam como deve ser o processo de desenvolvimento de *software*.

De acordo com Sommerville em (SOMMERVILLE, I. 2007, p.5), "Engenharia de software é uma disciplina de engenharia relacionada a todos os aspectos de produção de software*",* ou seja, ela abrange todas as etapas de desenvolvimento de *software*, através de teorias, regras, ferramentas e métodos que irão auxiliar na produção.

Através dessas metodologias pretende-se chegar ao "*software* ideal", ou seja, realizar a produção de uma ferramenta que atenda aos interesses tanto dos desenvolvedores quanto dos clientes, sem que ocorra eventuais imprevistos. E para que essas metodologias sejam seguidas algumas bibliografias como (SOMMERVILLE, I. 2007) e (PRESSMAN, R. 1997) esboçam possíveis padrões a serem seguidos.

Essa padronização vem sendo seguida em algumas equipes de desenvolvimento, onde cada um recebe uma função ou cargo, tornando responsável por determinada etapa do processo de desenvolvimento e assim dando origem a uma equipe. Entretanto, de acordo com (A Catedral e o Bazar Eric S. Raymond,) essa padronização não é a única solução para o sucesso de um software, pode-se alcançar o sucesso através de colaborações de pessoas que não tenham qualquer função determinada ou que não tenham que cumprir com prazos ou normas, método que o autor denomina como Bazar. Verifica-se o uso dessa conduta principalmente nos projetos open source (FUGGETA, A. 2003), que em alguns projetos foi fundamental para garantir o sucesso, como por exemplo, o GNU/Linux, o Apache e o Mozilla.

Mediante o sucesso dos projetos open source que utilizam a metodologia Bazar para o desenvolvimento, este trabalho irá descrever como se dá esse processo de desenvolvimento ajudando na compreensão dessa metodologia.

1.1 Motivação

Verifica-se que nos últimos anos houve um aumento no número de publicações que buscam fornecer alternativas que demonstrem como as equipes voltadas para projetos *open source*, conseguem alcançar o sucesso, porém poucos conseguem abordar o assunto sobre o modelo e metodologia que são aplicados.

1.2 Objetivos

O objetivo desse trabalho é mostrar através do levantamento das principais características dos projetos *open source* como alguns desses projetos conseguem obter sucesso fugindo dos processos considerados como clássicos por alguns autores.

Dessa forma, além de mostrar como esse modelo pode ajudar no gerenciamento desses projetos, mostram umas das formas de como ocorre o gerenciamento de alguns projetos *open source*.

1.3 Estrutura do Trabalho

No capítulo 2 é realizada uma breve descrição de alguns processos de software clássicos da literatura.

O capítulo 3 é destinado à explicação do termo *open source* e suas principais características.

O capitulo 4 mostra o gerenciamento de projetos *open source*, através da exposição de ideias de alguns estudiosos desse tema, além de fazer uma breve explicação dos modelos Catedral e Bazar e por último a conclusão deste trabalho.

Capítulo 2: Processos de Desenvolvimento de Software

Sommerville (SOMMERVILLE, I. 2007, p.5), descreve processo de *software* como: "Um conjunto de atividades cujo objetivo é o desenvolvimento ou a evolução do software*",* esse conjunto é composto por atividades consideradas fundamentais para o desenvolvimento do *software,* das quais se destacam:

- Especificação: atividade destinada a descobrir e definir as funcionalidades e limitações que existirão no *software,* em alguns casos há a participação dos clientes para definir as funcionalidades.
- Desenvolvimento: atividade destinada à programação do *software*, ou seja, inicia o desenvolvimento do código fonte através de implementações já determinadas de acordo com as especificações que já foram definidas.
- Validação e verificação: atividade destinada para verificar se as funcionalidades do *software* estão de acordo com o especificado nos processos, para revisar as especificações.
- Documentação: atividade realizada para documentar as funcionalidades, código fonte e possíveis alterações no decorrer do desenvolvimento.

2.1 Modelos de Processos de Software

Existem alguns modelos elaborados por alguns autores como Winston Royce em (ROYCE, W., 1970. p. 1–9.), que propõe o modelo cascata. Nesse modelo fica claro as divisões entre as atividades envolvidas no desenvolvimento do projeto. A Figura 2 ilustra claramente o modelo cascata.

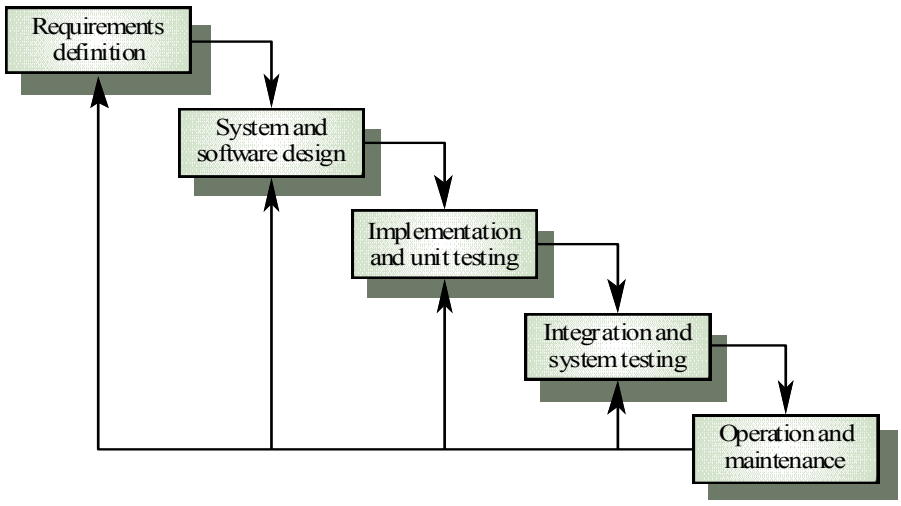

Figura 1. Diagrama do ciclo de vida cascata de Winston Royce na década de 70.

Já no modelo evolucionário, não há uma sequência linear, pois nesse modelo o *software* é produzido e lançado gradualmente, sendo assim, para cada ciclo de lançamento realiza-se as atividades de acordo com as necessidades do projeto. Sommerville em (SOMMERVILLE, I. 2007, p.45), sugere que o modelo evolucionário torna-se mais eficaz que o cascata, pois no modelo evolucionário a especificação poderá ser desenvolvida de forma incremental, conforme as necessidades dos usuários, porém esse modelo tem seus problemas, porque em alguns casos podem produzir sistemas com problemas estruturais. A Figura 2 (SOMMERVILLE, I. 2007, p.46) ilustra esse modelo.

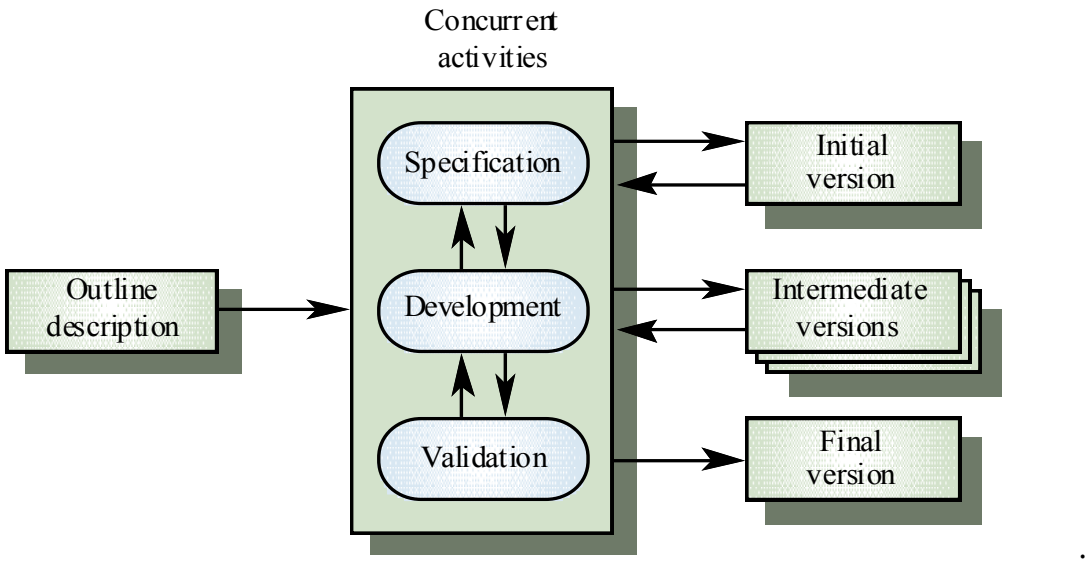

Figura 2. Modelo de desenvolvimento evolucionário.

Outro modelo conhecido é o modelo de sequência, pois determina uma sequência linear das atividades do projeto, de acordo com esse modelo trabalha-se primeiro com os requisitos para depois passar para codificação, testes e assim por diante. Entretanto é um modelo que dificilmente tende a ser seguido, pois raramente surgem *software* desenvolvidos nesse modelo, porque é muito difícil seguir uma sequência linear das tarefas envolvidas no projeto.

Capítulo 3: Processo de Desenvolvimento de Software Open Source

3.1 Definição

A discussão de sobre software open source inicia-se já na própria definição, pois seu sentido pode variar de acordo com determinados contextos, porém não é o foco desse trabalho gerar discussões filosóficas.

De acordo com Fuggetta, *open source* pode ser definido como:

"qualquer software cuja licença garante ao seu usuário liberdade relacionada ao uso, alteração e redistribuição. Seu aspecto fundamental é o fato do código-fonte estar livremente disponível para ser lido, estudado ou modificado por qualquer pessoa" (Fuggetta, A.,2003, p.77), Dessa forma, ele define em poucas palavras a complexidade que existe em torno da definição desse termo.

3.2 Características dos Projetos *Open Source*

Embora existam várias discussões filosóficas em torno dos projetos open source, ele possui características bem definidas pela comunidade que o torna diferente dos projetos convencionais, essas características são:

- Planejamento: em virtude das atividades serem realizadas por diversos colaboradores de diferentes países, nem sempre é possível realizar planejamento das atividades.
- Desenvolvimento voluntário: Alguns desenvolvedores realizam trabalhos sem receber qualquer ajuda financeira, para essas pessoas, a recompensa vem através do aprendizado, interesses por novos desafios e o prazer de compartilhar o conhecimento na comunidade.
- Flexibilidade na escolha da função: não existe uma regra que determine a função do colaborador, o que existe nesse caso é bom senso entre os colaboradores, dessa forma, na maioria das vezes colaboradores iniciantes são designados para exercerem funções mais básicas, porém nada impede que um usuário mais experiente entre na comunidade gerenciando um projeto ou realizando programação avançada.

- Não existe horário determinado: nesse caso, o próprio colaborador realiza e define a jornada de trabalho, alguns colaboradores já possuem um emprego fixo e colaboram nas horas de folga, além disso, deve-se levar em conta o grande número de colaboradores que moram em diferentes países de diferentes fusos, culturas e etnias.
- Lançamento constante de versões: por ter um trabalho voluntário e sem compromisso com prazo ou normas contratuais, o *software* é lançado quando a equipe determina que já seja uma versão funcional, solucionar seus erros e realizar melhorias ocorrem com o passar do tempo.
- Usuário colaborador: o próprio usuário do *software* pode sugerir melhorias e ideias para a comunidade, além disso, ele pode tornar-se um colaborador em qualquer momento.
- Distribuição eletrônica ou por mídia: a distribuição pode ser realizada através da *internet:* por *e-mail*, *sites* ou mídias (CD±R ou DVD±R).
- Descentralização: não há um local físico especifico para reunir os colaboradores, dessa forma em alguns casos as discussões e sugestões são encaminhadas para uma lista de e-mail ou fóruns de discussões voltados à comunidade.

3.3 Organização

Embora exista uma flexibilidade em determinadas características desse projeto, pode-se definir uma mínima organização de como ocorre o desenvolvimento do *software*. A Figura3 (REIS, C., 2001 p. 15) ilustra como funciona a organização de projetos *open source.*

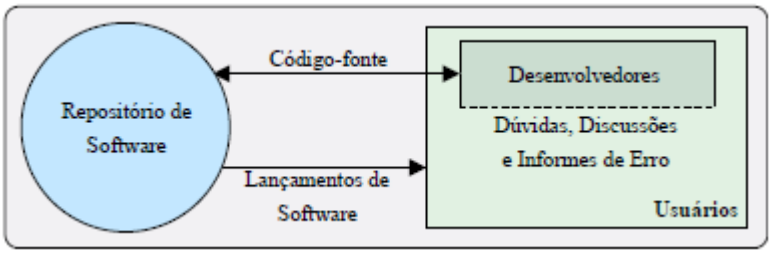

Figura 3. Possível diagrama de um projeto *open source*.

Observa-se que há um repositório onde todos os códigos-fontes são armazenados e que os desenvolvedores pertencem ao grupo de usuários, pois conforme explicado anteriormente, nesse tipo de projeto os usuários podem exercer a função de desenvolvedores também. O repositório é mantido através de ferramentas existentes na internet e possui um controle rigoroso de versões através de ferramentas especificas para o controle de versão, um exemplo é o CVS através do auxilio dessa ferramenta é possível que vários colaboradores trabalhem com o mesmo código-fonte. Além disso, as funcionalidades do CVS permitem que os colaboradores não precisem ficar conectados a *internet* enquanto realizam as correções ou aprimoramento do código-fonte, sendo possível que o colaborador realize as alterações em uma cópia local.

As alterações realizadas nos arquivos serão notificadas no repositório onde cada arquivo alterado receberá uma identificação numérica, evitando assim eventuais confusões no repositório, além disso, o CVS permite que ocorra facilmente a comparação entre as alterações realizadas nas cópias locais com qualquer outra versão que esteja no repositório *on-line*.

Outro fator muito importante na organização de projetos *open source* são os eventuais "papéis" que os colaboradores envolvidos no projeto exercem. Esses "papéis" podem ser definidos como:

- Usuários participativos: são usuários que além de usarem o *software* desenvolvido pela comunidade, também contribuem para o projeto, através da informação de erros e de sugestões para melhorar ou implementar novas funcionalidades no *software*. Dessa forma esses usuários acabam exercendo a função de testadores do *software*.
- Usuários não participativos: são usuários que não manifestam qualquer interesse em discutir, opinar ou ajudar no desenvolvimento do *software*, podem até encontrar erros ou desejarem modificações, porém não informam, apenas usam o software.
- Desenvolvedores casuais: são usuários com conhecimentos em desenvolvimento de *software* e capazes de realizar alterações no código fonte, suas alterações em alguns casos são alterações de caráter pessoal, ou seja, para ajudar em problemas que estão no dia-dia pessoal.

- Desenvolvedores ativos: são pessoas que tem participação constante nos projetos, nesse caso verifica-se uma possível responsabilidade com a comunidade, pois suas alterações no código fonte são mais complexas e extensas.

Embora exista essa classificação de alguns "papéis", não se verifica nenhuma norma ou regra que proíba a troca de "papéis" entre os colaboradores, muito pelo contrário, essa troca não só é permitida como é uma das diferenciações existentes nesse tipo de projeto em relação aos demais. Como nem todos os colaboradores se encontram no mesmo território, a marca registrada para comunicação entre eles é a *internet*, pois será através dela que muitos dos colaboradores irão se comunicar, discutir, opinar e sugerir novas ferramentas. A comunicação pode ser realizada através de e-mails, lista de discussões e fóruns da comunidade, dessa forma elimina as barreiras geográficas, culturais e étnicas. Alguns desses colaboradores dificilmente se encontrarão pessoalmente, pois o foco nesse caso não é reunir todos em um único ambiente e sim possibilitar que cada colaborador possa contribuir por livre e espontânea vontade.

Outra característica marcante nesse tipo de projeto é a disponibilidade do *software* para os usuários. O acesso pode vir através de *download* realizado em *sites* da própria comunidade ou através da distribuição direta entre os colaboradores o que de certa forma, ajuda a popularizar os aplicativos e ao mesmo tempo torná-lo o mais acessível possível, além disso, quanto maior o número de pessoas que utilizem o *software* e contribua para a comunidade, maior será o sucesso do projeto.

Capítulo 4. Gerenciamento de Projetos Open Source

Os projetos realizados na forma clássica ou Catedral conforme definição de Raymond em (RAYMOND, E. S., 1999. 27–78 p.), são gerenciáveis através de padrões pré-estabelecidos de acordo com suas necessidades, sempre respeitando um padrão hierárquico, onde os integrantes já possuem um "papel" definido e que prazos, cronogramas e metas devem ser cumpridas com rigor. É inegável que essas metodologias clássicas consigam sucesso, porém a metodologia empregada nos projetos *open source* são totalmente diferentes, pois não há preocupação com prazos, os "papéis" são definidos pelos próprios integrantes, flexibilidade nos horários de trabalho, em alguns casos nenhum tipo de comprometimento ou remuneração financeira e mesmo assim verifica-se um grande sucesso nesses projetos.

Raymond não só enfatiza isso, como também (RAYMOND, E. S., 1999.), comenta sobre as características de gerenciamento de projetos *open source,* compartilha sua experiência pessoal ao elaborar um projeto seguindo esses moldes de produção.

4.1 Metodologia Bazar

Raymmond em (RAYMOND, E. S., 1999.), descreve o modelo Bazar como uma forma de trabalho descentralizada, através da participação de voluntários e sem preocupação com prazos. Nesse modelo, a participação é de pessoas voluntárias, que não recebem nenhum auxílio financeiro e estão no projeto apenas para cooperar e compartilhar seus conhecimentos.

Nesse modelo não há uma definição formal dos "papéis", é um modelo totalmente informal, não há prazos, há trocas constante de responsabilidades, as versões do *software* são disponibilizadas constantemente e os próprios membros utilizam e já relatam os erros e sugestões para serem implementadas.

4.2 Qualidade

Na parte de qualidade, os colaboradores designados para essa tarefa, terá como responsabilidade controlar a qualidade do software. Esse controle de qualidade inclui identificar problemas em geral no projeto, revisar e exercer uma auditoria.

Caso seja uma pessoa seja responsável por revisar o código, ela poderá exercer os seguintes "papéis":

- Moderadores: são os colaboradores responsáveis dela gerencia de revisão;
- Revisores: colaboradores responsáveis por encontrar os erros existentes no código fonte;
- Documentadores: colaboradores responsáveis por documentar e registrar os resultados obtidos na revisão do código fonte;

A realização constante desse controle de qualidade muito provavelmente resultará na produção de *software* cada vez mais confiável.

4.3 Desenvolvimento

Nesta parte do projeto encontra-se a fase de produção do código fonte, as correções de erros e implementações de sugestões viáveis. Além disso, considera-se esta fase um pouco formal, pois cada tempo de desenvolvimento varia de acordo com o projeto.

4.4 Especificar Requisitos

Em alguns casos, *software open source* não se inicia depois da especificação formal de requisitos, e sim na fase de desenvolvimento. Os projetos que seguem as especificações de requisitos tem uma vantagem considerada, pois os desenvolvedores já tem uma noção sobre as funcionalidades do *software* antes de saírem programando.

O ideal nessa fase é documentar todas as alterações realizadas no código fonte de forma que outro desenvolvedor que acesse essas alterações, consiga identificar onde ocorreram as alterações e, além disso, deixar justificado o motivo de cada alteração.

4.5 Lançamento de versões

Essa fase do processo depende do colaborador que foi designado como gerente de *release,* nesse caso, ele terá a função de construir módulos dos projetos e realizar o controle de versão, através disso é possível encontrar os erros existente e que precisam ser solucionados para que depois consiga disponibilizar uma versão que apresente as funcionalidades essências para a viabilidade do *software*.

Depois se deve disponibilizar a versão para a comunidade, através de *sites,* servidores e mídias. E conforme usuários venham a utilizar o *software e* detectar possíveis erros.

Conclusões

Em virtude da grande popularização e aceitação dos projetos *open source* (RAYMOND, E. S., 1999 p 2.) percebe-se que embora não funcione de acordo com as maiorias dos padrões clássicos sugeridos para desenvolvimento, não deixa de ser um grupo de *software* de qualidade, e que o principal fator para essa popularização é a forma de como reunir várias pessoas de diferentes localidades a colaborarem, disponibilizar seus conhecimentos e idéias através de listas de e-mails. Verifica-se que em alguns casos há pessoas que realizam colaborações em suas horas vagas e que não se preocupa com retorno financeiro, apenas em ajudar demais pessoas, ou em alguns situações apenas para promover desafios pessoais.

Por ser um projeto que envolva pessoas com diferentes problemas, projetos *open source* podem possuir um processo de evolução constante, pois para alguns colaboradores determinada versões já atenderem as suas necessidades até o momento e para outros ainda precise de aperfeiçoamentos, e nada impede que essas incrementações continuem.

Sendo assim, pode-se definir que se trata de um estilo de desenvolvimento que valoriza principalmente a participação ativa de seus colaboradores, pois e através dessa atuações que erros, sugestões e manutenção do *software* seja realizada rapidamente

Referências Bibliográficas

SOMMERVILLE, I. **Engenharia de Software.** 8th ed., Addison-Wesley, 2007. 5 – 7 p.

SOMMERVILLE, I. **Engenharia de Software.** 8th ed., Addison-Wesley, 2007.

PRESSMAN, R. S. **Software Engineering: A practitioner's approach.** 4th. ed. McGraw-Hill,1997. 22–53 p.

FUGGETA, A. **Open source software - an evaluation., Journal of Systems and Software**. 77-90 p.

SOMMERVILLE, I. **Engenharia de Software.** 8th ed., Addison-Wesley, 2007. 5 – 7 p.

ROYCE, W. W. **Managing the development of large software systems.** In: *Proceedings of IEEE WESCON*. [S.l.: s.n.], 1970. p. 1–9.

SOMMERVILLE, I. **Engenharia de Software.** 8th ed., Addison-Wesley, 2007.

SOMMERVILLE, I. **Engenharia de Software.** 8th ed., Addison-Wesley, 2007.45 p.

FUGGETA, A. **Open source software - an evaluation., Journal of Systems and Software**. 77-90 p.

SOMMERVILLE, I. **Engenharia de Software.** 8th ed., Addison-Wesley, 2007.
CHRISTIAN, R. **Caracterização de um Modelo de Processo para Projetos de Software**, 2001. 15 p.

RAYMOND, E. S. **The Cathedral and The Bazaar. In: The Cathedral and The Bazaar. 1st ed. Sebastopol**: O'Reilly and Associates, 1999. 27–78 p.

RAYMOND, E. S. **The Cathedral and The Bazaar. In: The Cathedral and The Bazaar. 1st ed. Sebastopol**: O'Reilly and Associates, 1999. 2 p.