Desenvolvimento de Software Livre

Aspectos Culturais e Tecnológicos

Prof. Dr. Fabio Kon e Nelson Lago

Centro de Competência em Software Livre

IME-USP

13/02/2009 - SERPRO

Por que Software Livre?

- **Diversas razões para adotar software livre:**
  - Software livre como opção ética
  - Software livre e seu impacto na sociedade do futuro
  - Software livre como metodologia tecnicamente interessante - nosso foco aqui, mas não a razão mais importante!

Conteúdo

- **Rápida revisão sobre o modelo de software livre e as comunidades envolvidas**
- **Licenças de software livre e aspectos legais**
- **Processo de desenvolvimento colaborativo**
  - Relação com a comunidade
  - Usuários e desenvolvedores externos
- **Boas práticas de desenvolvimento**
- **Ferramentas de suporte mais comuns**

Conteúdo

- **Rápida revisão sobre o modelo de software livre e as comunidades envolvidas**
- **Licenças de software livre e aspectos legais**
- **Processo de desenvolvimento colaborativo**
  - Relação com a comunidade
  - Usuários e desenvolvedores externos
- **Boas práticas de desenvolvimento**
- **Ferramentas de suporte mais comuns**

Por que Software Livre?

- **Software livre permite o compartilhamento de código, otimizando o uso dos recursos**
  - Menos duplicação de esforço
  - Menor custo de desenvolvimento
- **Oportunidade para melhoria da qualidade**
  - Vários olhos enxergam mais
  - Orgulho pessoal incentiva desenvolvedor a ser mais cuidadoso
  - Vários usuários envolvidos promovem melhorias e relatórios de erros
- **O mercado de desenvolvimento de software livre é um mercado local, interessante para o Brasil**

Mas...

- **Não basta simplesmente liberar o código**
  - SAP DB (MaxDB) entre 2001 e 2004
- **É preciso também**
  - Fomentar a criação de uma Comunidade
  - Interagir e colaborar com a Comunidade
  - Garantir que a colaboração seja bem organizada

- **Sem a Comunidade**
  - Os maiores benefícios do SL não serão obtidos
- **Com Comunidade mal gerida**
  - Pode trazer mais problemas que vantagens

O que é Software Livre

- **Comunidade de software livre é antiga**
  - Originalmente informal, depois formal (através da *Free Software Foundation*)
- **Compartilhamento do código-fonte e troca de idéias**
- **Só é possível em um ambiente que facilite a troca de código-fonte**
  - Por isso, o crescimento junto com a Internet
- **Explorar o software livre comercialmente envolve trabalhar com a comunidade**

Comunidades

- **Diversos tipos de pessoas e entidades, com diferentes interesses e pontos de vista, estão envolvidos com software livre**
  - Software Livre – *Free Software Foundation*
  - *Open Source* (Fonte Aberto) – OSI
  - Pragmáticos – várias empresas
  - “Radicais”

Conteúdo

- Rápida revisão sobre o modelo de software livre e as comunidades envolvidas
- **Licenças de software livre e aspectos legais**
- Processo de desenvolvimento colaborativo
  - Relação com a comunidade
  - Usuários e desenvolvedores externos
- Boas práticas de desenvolvimento
- Ferramentas de suporte mais comuns

Tipos de licenças

- **Diferentes visões sobre o modelo ideal para o software livre, que se traduz em diferentes licenças**
- **3 tipos principais de licenças:**
  - Recíprocas totais (GPL e assemelhadas): o software é livre, deve permanecer livre e trabalhos derivados devem ser também livres
  - Recíprocas parciais (LGPL, MPL e assemelhadas): o software é livre e deve permanecer livre, mas trabalhos derivados não precisam ser livres
  - Permissivas (Apache, MIT/X11, BSD e assemelhadas): o software é livre, mas pode ser relicenciado sem permissão adicional do autor

A licença GPL

- **60% do código de uma distribuição típica é GPL**
  - A segunda licença mais popular é a LGPL, com 7%
- **GPL só versa sobre a distribuição; qualquer uso é permitido, incluindo combinações com software fechado**
- **Por ser uma licença, não depende de assinatura**
- **Violar a GPL é violar a lei de *Copyright***
  - É impossível distribuir código GPL legalmente sem aceitar a GPL, independentemente de sua "validade legal"

Compatibilidade entre licenças

- **Diferentes licenças impõem diferentes condições**
- **Problemas de compatibilidade são comuns com a GPL, por causa do mecanismo de *copyleft***
- **Muitas vezes, detalhes legais, como cláusulas que definem um foro específico para resolução de conflitos**
- **OSI classifica várias licenças explicitamente como "redundantes"**
  - Muitas são equivalentes em intenção, mas ainda assim incompatíveis
- **Solaris e Linux não podem usar código um do outro por incompatibilidade entre as licenças**

Compatibilidade entre licenças

- **Identificar compatibilidade ou não entre licenças é complexo e pode haver impacto legal**
  - Vale muito a pena ser compatível com a GPL
    - Facilidade para agregar código alheio
    - Dificilmente há boa razão *prática* para não ser compatível
    - Compatibilidade pode ser de “mão-única” (como no caso do FreeBSD X Linux)
- **Problemas podem ser sutis**
  - O uso da Qt (não-livre na época) pelo KDE
    - Teoricamente, KDE só pode ser distribuído sob a GPL3
  - openssh: código livre é livre para sempre
  - Mplayer e sistemas embarcados
  - “Tivoization”

Conteúdo

- Rápida revisão sobre o modelo de software livre e as comunidades envolvidas
- licenças de software livre e aspectos legais
- **Processo de desenvolvimento colaborativo**
  - Relação com a comunidade
  - Usuários e desenvolvedores externos
- Boas práticas de desenvolvimento
- Ferramentas de suporte mais comuns

O processo de desenvolvimento de software livre

- **Em projetos de software livre, os papéis de desenvolvedor, usuário, gerente de projeto, muitas vezes se confundem**
  - Só é possível liderar quando outros estão dispostos a seguir
  - As questões éticas são importantes para a comunidade
  - As questões técnicas *também* são importantes para a comunidade
  - Planos, decisões, metodologias e aspectos técnicos precisam ser negociados
  - Mudanças de rumo imprevistas podem ocorrer e devem ser encaradas com naturalidade

O processo de desenvolvimento de software livre

- **Alguns princípios de "A catedral e o bazar":**
  - Programas nascem (e se desenvolvem) por necessidades pessoais (*scratch an itch*)
  - Escrever bom código vs. reutilizar bom código
  - Usuários são co-desenvolvedores
  - Distribuir logo e com frequência (*release early, release often*)
  - Com vários olhos, todo bug é evidente (*given enough eyeballs, all bugs are shallow*)
  - Coordenar contribuições é fundamental

Aspectos práticos da relação com a comunidade

- **É importante garantir a motivação dos envolvidos**
  - Senso de comunidade
- **A entrada de novos contribuidores deve ser facilitada**
- **A saída abrupta de contribuidores não deve ser um problema**

Aspectos práticos da relação com a comunidade

- ***Patches*, relatos de erros e outras contribuições externas devem ser recebidos com atenção e retorno rápido**
  - Notificar usuário quando um relato de erro foi recebido
  - Informar usuário quando o erro foi corrigido
  - Decidir rapidamente sobre se e quando incorporar uma contribuição externa e responder ao colaborador
  - Responder dúvidas de colaboradores externos com agilidade
  - O usuário é seu amigo!

Conteúdo

- Rápida revisão sobre o modelo de software livre e as comunidades envolvidas
- licenças de software livre e aspectos legais
- Processo de desenvolvimento colaborativo
  - Relação com a comunidade
  - Usuários e desenvolvedores externos
- **Boas práticas de desenvolvimento**
- Ferramentas de suporte mais comuns

O código no dia-a-dia

- **O código precisa ser claro e bem modularizado**
  - Facilita a leitura por novos contribuidores e por eventuais contribuidores
  - Facilita que mudanças sejam feitas sem conhecimento completo do código
  - **Testes automatizados** facilitam alterações
  - Nenhum código é intocável: refatoração deve ser rotineira
  - Integração contínua é fundamental, caso contrário nenhum contribuidor externo será capaz de participar do processo (*release early, release often*)

O código no dia-a-dia

- **A documentação não precisa ser extensa, mas precisa estar correta e atualizada**
  - A utilidade principal da documentação é facilitar a vida dos novos contribuidores
  - Formalidade, explicações referentes a processos internos de definição de características, etc. não são relevantes
  - Melhor qualidade que quantidade

O código no dia-a-dia

- **Propriedade coletiva do código**
  - Todos são responsáveis por tudo
  - Todos conhecem pelo menos superficialmente tudo
  - Sem entraves burocráticos para modificar qualquer parte do código
- ***"Show me the code"***
  - A maneira mais fácil de acabar com discussões intermináveis
  - Diferentes soluções podem ser experimentadas na prática

Conteúdo

- Rápida revisão sobre o modelo de software livre e as comunidades envolvidas
- licenças de software livre e aspectos legais
- Processo de desenvolvimento colaborativo
  - Relação com a comunidade
  - Usuários e desenvolvedores externos
- Boas práticas de desenvolvimento
- **Ferramentas de suporte mais comuns**

Ferramentas de apoio: gerência de relatos de erros

- **Gerenciadores de erros/problemas/solicitações *Bug trackers* (bugzilla, trac)**
  - *Feedback* automático
  - Acompanhamento e histórico
  - Visão geral do estado do projeto
  - Facilidade em reportar novos erros
  - Facilidade em identificar erros repetidos
  - Garante que um erro não será esquecido ou abandonado

Ferramentas de apoio: meios de comunicação

- **Comunicação e documentação**
  - Listas de discussão
    - Latência
    - Todos leem
    - Histórico, inclusive na Web
    - Bom para “grandes discussões”
  - IRC
    - Imediato
    - Sem histórico útil
    - Bom para resolver dúvidas rápidas
  - Wiki
    - Solução simples para manutenção da documentação
  - Fóruns
    - Pouco usados, mas similares às listas de discussão

Ferramentas de apoio: gerência de tarefas e projetos

- **XPlanner**
  - Usado localmente no CCSL com grande sucesso

XPlanner

Top | Project Content: Search | ID: Find | Integrations | Me

Iteration Versão 0.2 (2008-12-09 to 2009-01-24) [id=94363]

« SaguiSaúde »
« Versão 0.2 »

Esta versão tem como objetivo a integração do Borboleta com o Sagui.

Hours: Estimate 284,0, Actual 145,0, Remaining 139,0

Save order

Actions	ID	Order	User Story	!	Cust.	Progress	Act.	Rem.	Cur. Est.	Orig. Est.	Tasks	Tracker	Disp.	Status
	94493	1	Troca de mensagens entre Borboleta e Sagui	1	hd		39,0	53,0	92,0	76,0	6	GLD	Planned	Defined
	94496	2	Módulo Borboleta no Sagui	1	JRB		77,0	56,0	133,0	126,0	3	TC	Planned	Defined
	94495	3	Login autenticado	2	RK		25,0	6,0	31,0	32,0	4	hd	Planned	Defined
	94494	4	Importação de dados do FoxPro	2	SCF		4,0	24,0	28,0	40,0	3	hd	Planned	Defined

Save order

Edit | Delete | Create Story | Close | Import | Export | Stories | All Tasks | Metrics | Charts | Accuracy | History | Print

Notes: Add Note/Attachment

user: kon [Logout] Version 0.7b7 built 05/24/2006 (rev 1149)

Contact your Administrator for production help. Any enhancement or bug report should be created following these instructions. Click here to get your system information.

Ferramentas de apoio: controle de versões

- **Gerência de versões**
  - CVS
    - Vale muito a pena migrar para subversion
  - Subversion (svn)
    - Estável, rápido e versátil
    - Muito usado

Ferramentas de apoio: controle de versões

- **Mas a "nova onda" são os sistemas descentralizados**
  - Mais fácil realizar integração (*merges*)
  - Mais fácil manter versões experimentais do código (*branches*)
  - Mais fácil incorporar alterações de colaboradores fora do time principal
  - Mais fácil "escolher arroz"
  - Melhor manutenção do histórico

Ferramentas de apoio: controle de versões

- **Fluxo de trabalho ligeiramente diferente**
  - Cada desenvolvedor tem uma cópia completa do repositório
  - *Commits* são realizados localmente, guardando histórico individual
  - Uma mesma alteração pode ser mesclada múltiplas vezes
  - A cada operação de mescla, a operação é registrada como um único *commit*, mas os commits individuais que a compõe continuam no histórico como *sub-commits*
  - Desenvolvedores podem trabalhar sem acesso de escrita ao repositório principal

Ferramentas de apoio: controle de versões

- **Git**
  - Desenvolvido originalmente por Linus Torvalds
  - Usado no kernel do Linux, Ruby on rails e outros
  - O mais rápido para repositórios grandes
- **Bazaar**
  - Desenvolvido pela canonical (ubuntu)
  - Usado pela canonical (launchpad, ubuntu) e mySQL
  - Mais funcionalidades e mais flexível
- **Mercurial**
  - Desenvolvido independentemente
  - Usado pelos projetos da Sun (OpenSolaris, OpenJDK)
  - Bom desempenho, boas funcionalidades

Ferramentas de apoio: hospedagem

- **Sourceforge e GForge**
  - Hospedagem de projetos com CVS e SVN
  - Código fonte disponível
- **GitHub**
  - Hospedagem de projetos com Git
  - Usado pelo Ruby on rails
- **Launchpad**
  - Hospedagem de projetos com bazaar
  - Usado pelo Ubuntu e outros
  - Pode funcionar como “meta-hospedagem”
  - Código será aberto em junho

Conclusão

- **O modelo aberto de desenvolvimento pode trazer muitos benefícios**
  - Qualidade
  - Agilidade
  - Aceitação
- **Mas isso só funciona se há uma comunidade**
  - participativa e
  - bem organizada.
- **A escolha de boas ferramentas de apoio pode ajudar significativamente (mas não é tudo)**

O CCSL

- **Centro de Competência em Software Livre do IME/USP**
  - Membro da rede internacional QUALIPSO
  - http://ccsl.ime.usp.br

  - Pesquisa e desenvolvimento de SL
  - Pesquisa sobre o ecossistema do SL

  - Palestras, seminários, eventos
  - Cursos
  - Assessoria