Qualipso
Quality Platform for Open Source Software

Uso de software livre na indústria

oportunidades e modelos de negócio

Nelson Lago
CCSL IME-USP

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 base para negócios – o foco desta aula (**mas não o mais importante!**)
  - Do ponto de vista do mercado, as duas primeiras razões são semelhantes ao que existe hoje em relação à ecologia em processos industriais

Problemas do software restrito

- **Software é extremamente complexo**
  - Problemas com a qualidade
  - Múltiplos fornecedores duplicam esforços para oferecer soluções similares sem possibilidade de compartilhamento, como no caso de produtos físicos
  - Patentes são efetivamente irrelevantes
- **Cópias ilegais são um problema para fornecedores e usuários**
  - Para fornecedores, representam ameaça constante
  - Para usuários, trazem dificuldades de conformidade e problemas com sistemas anti-cópia (BSA)

Problemas do software restrito

- **Modelo de negócio baseado em software "de prateleira" não é funcional**
  - Software ≠ objeto; modelo cria dificuldades artificiais e não tira benefícios das características específicas do software
  - A maior parte do dinheiro gasto e dos postos de trabalho em software são voltados para outras áreas (cerca de 80%)
  - Apenas 5% do custo de uma solução corporativa corresponde ao custo das licenças
  - No entanto, cada aplicação personalizada precisa ser desenvolvida do zero!
    - O custo de cada aplicação personalizada é alto mesmo quando já há soluções similares existentes

Solução: software livre

- **Software livre permite o compartilhamento de código, simplificando o desenvolvimento**
  - Menos duplicação de esforço
  - Menor custo de desenvolvimento
- **A qualidade cresce "naturalmente"**
  - Vários olhos enxergam mais
  - Orgulho pessoal incentiva desenvolvedor a ser mais cuidadoso
  - Vários usuários envolvidos promovem melhorias e relatórios de erros
- **O mercado de software livre é um mercado local, interessante para o Brasil**

Solução: software livre

- **Não há restrição de fornecedor: mais interessante para o usuário**
- **Sempre é possível evoluir o código, mesmo que o fornecedor original abandone o mercado**
- **O modelo de negócio não é igual ao do software de prateleira**
  - 80% do dinheiro sem os problemas do software fechado
  - É preciso criatividade e várias abordagens
  - A competição é potencialmente mais acirrada
  - A reputação é fundamental

Problemas do software livre

- **Responsabilidade legal:**
  - licenças normalmente eximem o autor de responsabilidade, mas lei brasileira não permite
- **Qualidade:**
  - difícil avaliar qualidade entre as alternativas; nem sempre há uma instituição oferecendo garantias
- **Comprometimento:**
  - não há garantias de que um software será mantido e suportado no longo prazo (mas com software não-livre isso é ainda pior)
- **Sustentabilidade:**
  - se modelos de negócio tradicionais não funcionam, como garantir a sustentabilidade de um projeto?

Problemas do software livre

- **“Propriedade intelectual”:**
  - não há segredo industrial; concorrentes têm acesso ao código-fonte
- **Reputação e imagem:**
  - É difícil construir uma reputação frente à comunidade
  - software livre pode ter uma imagem problemática junto aos usuários (“se é gratuito, não pode ser bom”)
    - http://rn.softwarelivre.org/casada/?p=1
- **Pouca experiência da sociedade e do mercado em lidar com os modelos apropriados para software livre**

Problemas do software livre

- **Marcas registradas:**
  - software livre geralmente não trata de marcas registradas, e existem estratégias e problemas relacionados (firefox, redhat...)
- **Patentes:**
  - Brasil não tem patentes de software, mas isso pode mudar
  - restrições de patentes em outros países influenciam as decisões da comunidade
- **Mas é inevitável! Se você não adotar, seu concorrente vai**

Breve histórico

- **“Pré-história”:**
  - no meio comercial, software é uma pequena parte de uma solução muito maior, envolvendo hardware, consultoria etc.
  - no meio acadêmico, o software é compartilhado informalmente, como outras formas de conhecimento
- **1976 – Bill Gates e sua “carta aberta aos hobistas”**
  - Software para microcomputadores tem grande potencial comercial
  - Não é possível financiar desenvolvimento fora do modelo de prateleira

Breve histórico

- **1981 – IBM fecha acordo com Microsoft**
  - desconsidera a relevância do software e abre mão do copyright do DOS
  - expansão da Microsoft graças ao mercado de clones
- **1984 – Richard Stallman lança o projeto GNU**
  - uso de software restrito não é ético
- **1991 – Linus Torvalds propõe a criação do Linux**
  - projeto informal, apenas para estudo
- **1995 – Boom da Internet**
  - tem início a comercialização em escala do Linux (Red Hat etc.)

Breve histórico

- **1997 – Eric Raymond apresenta “*a catedral e o bazar*”**
  - vantagens técnicas do software livre
  - mecanismos de funcionamento do desenvolvimento descentralizado
- **1998 – Netscape libera o código fonte do navegador Mozilla sob licença livre**
- **1998 – Eric Raymond, Linus Torvalds e outros lançam o movimento *open source***
  - software livre por razões técnicas
  - expressão *open source* ao invés de *free software*

Breve histórico

- **1999 – Sourceforge é lançado**
  - no Brasil, Projeto Software Livre Brasil (PSL-BR)
- **2000 – OpenOffice é lançado**
  - no Brasil, primeiro Fórum Internacional de Software Livre (FISL)
- **2001 – IBM anuncia investimento de US$1bi no linux**
  - o brasileiro Marcelo Tosatti, com 18 anos, é escolhido por Linus Torvalds como mantenedor oficial da versão 2.4 do kernel do linux

Breve histórico

- **2003 – SCO processa IBM**
  - suposto código de sua propriedade inserido no linux
  - dúvidas no mercado sobre software livre e a GPL
  - para todos os efeitos práticos, perdeu
- **2005 – Sun lança Solaris 10 sob licença livre**
- **2006 – Protótipo do XO (*One Laptop per Child*)**
- **2007 – FSF lança GPL versão 3**
  - “tivoization”
- **2007 – Sun distribui JDK sob a GPL versão 2**
  - no Brasil, GINGA-NCL é liberado
- **2008 – Nokia compra TrollTech**
  - Symbian será software livre

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
- **Portanto, explorar o software livre comercialmente só faz sentido se houver envolvimento da comunidade**
  - Essa é a grande vantagem!
  - Essa é a grande dificuldade!

Mas...

- **Não basta simplesmente liberar o código**
  - SAPDB
- **Sem a Comunidade**
  - os maiores benefícios do SL não serão obtidos
- **Com comunidade mal gerida**
  - pode trazer mais problemas que vantagens
- **Software Livre X software gratuito: o impacto do Java, Flash, Qt, drivers nVidia...**

As 4 liberdades

- **A FSF considera que um software é livre se oferece as 4 liberdades seguintes:**
  - Liberdade para executar o programa
  - Liberdade para estudar e modificar o programa
  - Liberdade para redistribuir o programa
  - Liberdade para melhorar e redistribuir as melhorias ao programa
- **A OSI e o projeto Debian usam definições bastante próximas da da FSF**

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
  - “nem aí” – Linus Torvalds
  - Pragmáticos – várias empresas
  - Radicais

A catedral e o bazar

- **Programas nascem de necessidades pessoais (*scratch an itch*)**
- **escrever bom código X 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**

Interação com a comunidade

- **Software livre depende da comunidade e do compartilhamento de ideias**
  - Não há hierarquia
  - Não há mecanismos de pressão
  - Não há muito formalismo no processo
- **Metodologias ágeis têm melhores chances de sucesso**
  - propriedade comunitária do código evita a dependência de pessoas específicas
  - desperdício de recursos é comum; mas esse desperdício não tem custo (*show me the code*)
    - Novas ideias podem ser discutidas e implementadas mais facilmente

Interação com a comunidade

- **É preciso entender as razões pessoais para o envolvimento da comunidade e incentivá-lo**
  - Em projetos de software livre, os papéis de desenvolvedor, usuário, gerente de projeto etc 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

Interação com a comunidade

- ***Patches*, relatos de erros e outra 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!

A cultura do desenvolvimento de software livre

- **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*)

A cultura do desenvolvimento de software livre

- **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

A cultura do desenvolvimento de software livre

- **Propriedade coletiva do código**
  - Todos são responsáveis por tudo
  - Todos conhecem pelo menos superficialmente tudo
  - Sem formalismos 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

Ferramentas

- **Bug trackers (bugzilla, trac)**
  - Feedback automático
  - acompanhamento e histórico
  - visão geral do status do projeto
  - facilidade em reportar novos bugs
  - facilidade em identificar bugs repetidos
  - garante que um bug não será esquecido ou abandonado

Ferramentas

- **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

- **xplanner**

Ferramentas

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

Ferramentas

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

Ferramentas

- **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

- **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 *features* e mais flexível
- **Mercurial**
  - Desenvolvido independentemente
  - Usado pelos projetos da Sun (OpenSolaris, OpenJDK etc)
  - Boa performance, boas features

Ferramentas

- **Sourceforge e GForge**
  - Hospedagem de projetos com CVS e SVN
  - Código fonte disponível, com ressalvas
- **GitHub**
  - Hospedagem de projetos com Git
  - Usado pelo Ruby on Rails
  - Não é livre
- **Launchpad**
  - Hospedagem de projetos com bazaar
  - Usado pelo ubuntu e outros
  - Pode funcionar como "meta-hospedagem"
  - Software livre

Licenças

- **3 tipos principais:**
  - Recíprocas totais (GPL e assemelhadas): o software é livre, deve permanecer livre e trabalhos derivados devem ser também livres
  - Recíprocas parciais (LGPL 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

Licenças – a 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 softwares restritos**
- **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**

Consequências da escolha do licenciamento

- **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”

Modelos de negócios

- **Software livre se tornou um fenômeno comercial a partir do final dos anos 90 e tem crescido**
- **Várias abordagens; algumas estão se tornando "tradicionais", mas há muito espaço para a criatividade**
- **Nichos específicos podem ser explorados por abordagens específicas**
- **Algumas abordagens são funcionais apenas para empresas de grande porte; outras, apenas para empresas de pequeno porte; e outras são mais versáteis**

Modelos de negócios

- **Redistribuição (CDs e DVDs com software livre)**
  - Dependendo do público-alvo e da disponibilidade da internet na região, pode ser muito interessante
- **Extensões não-livres**
  - Com base em um software livre, licenciamento não-livre de componentes adicionais
  - Usado pela IBM com o Eclipse, mas também por outros desenvolvedores não envolvidos com o núcleo
- **Produtos e serviços privilegiados**
  - Ao assumir uma posição de liderança no desenvolvimento de um software, possibilidade de oferecer serviços agregados, como a Sun e o Java

Modelos de negócios

- **Licenciamento duplo**
  - Disponibilização do software sob licença GPL para angariar usuários e desenvolvedores; relicenciamento remunerado sob licença não-livre, como a Qt
- **Licença com prazo de validade**
  - Novas versões podem ser liberadas sob licença não-livre enquanto versões mais antigas são relicenciadas sob licença livre, como o GhostScript
- **Serviços diretos**
  - Treinamento, suporte, integração, manutenção, personalização etc.

Modelos de negócios

- **Serviços padronizados**
  - Usuário paga uma assinatura pela manutenção de seu parque de máquinas, backups, serviço web etc., como ocorre com a RedHat e os provedores de hospedagem
- **Integração com produtos de hardware**
  - O software não é um produto, mas um mecanismo para auxiliar a venda do hardware, como a Nokia
- **Prestígio na comunidade**
  - Uma marca forte na comunidade e no mercado facilita a posição da empresa na oferta de serviços e consultoria, como ocorre com a RedHat

Modelos de negócios

- **Serviços baseados em software livre**
  - Oferta de serviços usando software livre, como o google ou provedores de acesso e hospedagem internet; pode ou não haver colaboração no desenvolvimento de acordo com seu interesse
- **Franquias**
  - Podem possibilitar a entrada de pequenas empresas no mercado utilizando recursos técnicos e de marketing de grande porte
- **Propaganda**
  - Em alguns casos, pode fazer sentido usar propagandas como fonte de renda, como ocorre com o firefox e o google ou o limewire

Casos de sucesso

- **A Sun**
  - JCP
  - Certificação
  - Problemas na migração da licença
  - Licenciamento duplo
  - Marca registrada “Java” é a garantia da unidade da plataforma
  - Outros produtos: GNOME, PostgreSQL, Solaris, OpenOffice

Casos de sucesso

- **A RedHat**
  - Marcas registradas, RHEL e CentOS
  - Serviços agregados (Red Carpet etc.)
  - Certificação e treinamento
  - 130mi/quarter de faturamento com assinaturas
  - 28mi/quarter de faturamento com treinamento e serviços

Casos de sucesso

- **MySQL**
  - Licenciamento duplo
  - Extensões não-livres
  - Consultoria
- **TrollTech**
  - Licenciamento duplo
  - Apoio ao KDE como forma de divulgar sua plataforma
- **Nokia**
  - Maemo é a infraestrutura para a venda de internet tablets e futuros telefones
- **CACE technologies**
  - Extensões não-livres sobre o wireshark

Casos de sucesso

- **IBM**
  - Fundações apache e eclipse: relicenciamento (WebSphere, Rational)
  - Consultoria, treinamento, integração, suporte...
- **Paggo**
  - Serviços baseados em software livre
- **Provedores de acesso e hospedagem**
  - Serviços baseados em software livre
  - serviços agregados
  - Consultoria
- **Impacta, 4Linux e outras**
  - Treinamento

Conclusão

- **Software livre veio para ficar**
  - "Quando não puder com seus inimigos, junte-se a eles"
- **O momento é de oportunidades**
  - o mercado ainda está longe de ser consolidado e há espaço para novas empresas, que podem assumir posições privilegiadas no longo prazo
- **Existem vários mecanismos para viabilizar a exploração econômica, mas é preciso criatividade**
- **É um caminho eticamente desejável**
- **O futuro é luminoso :)**