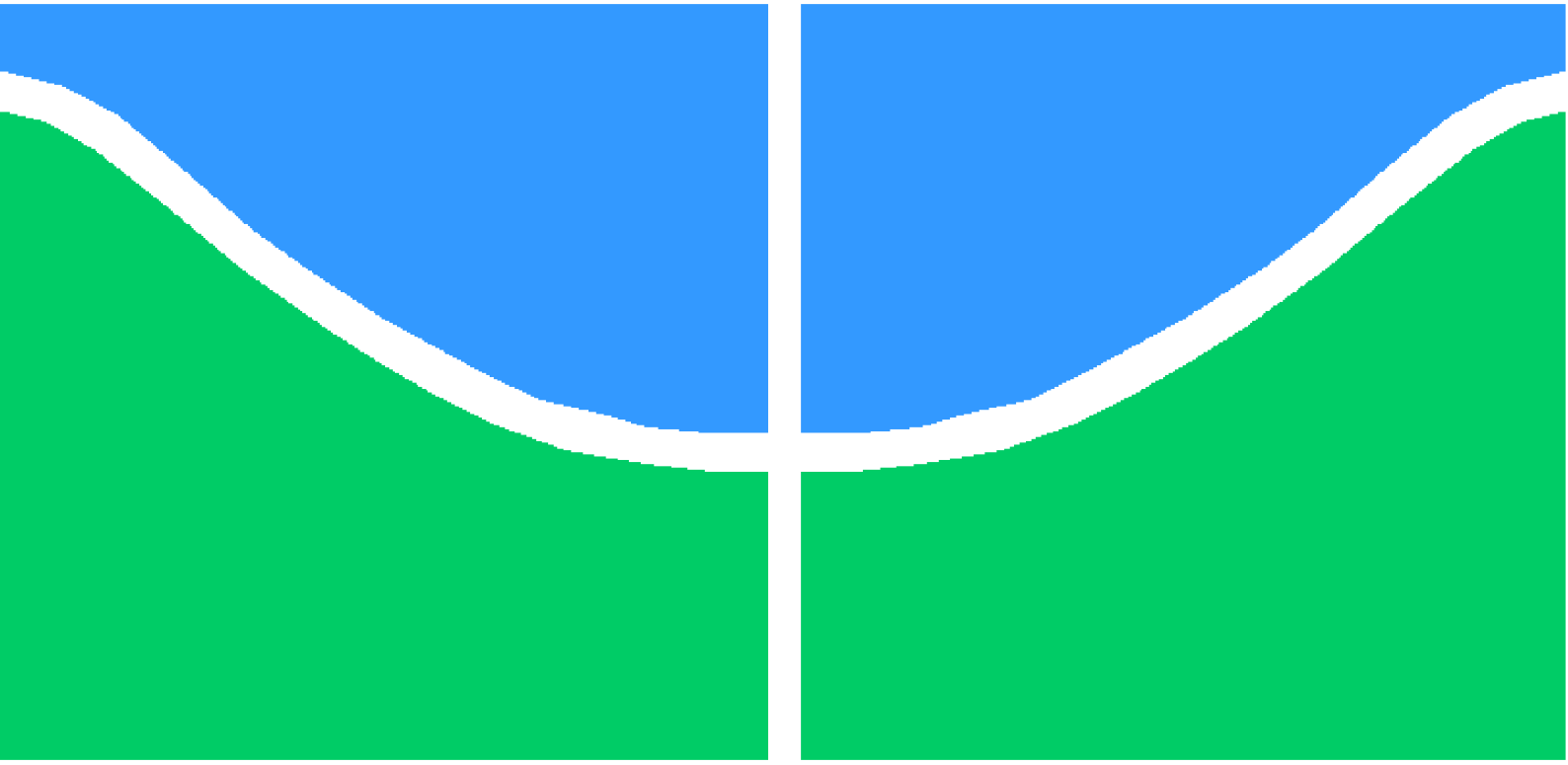

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

Evolução de Software: De Plugin para Aplicação Independente

Autor: Vinícius Vieira Meneses
Orientador: Prof. Dr. Paulo Roberto Miranda Meirelles

Brasília, DF
2013

Vinícius Vieira Meneses

Evolução de Software: De Plugin para Aplicação Independente

Monografia submetida ao curso de graduação em (Engenharia de Software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de Software).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Dr. Paulo Roberto Miranda Meirelles

Brasília, DF

2013

Vinícius Vieira Meneses
Evolução de Software:
De Plugin para Aplicação Independente / Vinícius Vieira Meneses. – Brasília, DF, 2013-
33 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Paulo Roberto Miranda Meirelles

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2013.

1. Evolução de Software. 2. Arquitetura de Software. I. Prof. Dr. Paulo Roberto Miranda Meirelles. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Evolução de Software:
De Plugin para Aplicação Independente

CDU 02:141:005.6

Vinícius Vieira Meneses

Evolução de Software: De Plugin para Aplicação Independente

Monografia submetida ao curso de graduação em (Engenharia de Software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de Software).

Trabalho aprovado. Brasília, DF, :

Prof. Dr. Paulo Roberto Miranda Meirelles
Orientador

Convidado 1

Convidado 2

Brasília, DF
2013

Resumo

[Inserir resumo... Exemplo Daniel]

O objetivo desse trabalho de conclusão de curso é apresentar uma justificativa teórica e levantar os requisitos necessários para viabilizar a implantação de uma rede de colaboração para a Universidade de Brasília que atue como um ambiente virtual para a criação e o compartilhamento de conhecimento de forma colaborativa e horizontal a ser utilizada como uma ferramenta complementar de apoio à educação. Esta abordagem permite quebrar a verticalização da relação professor-aluno e faz com que o aluno passe a atuar mais como autor ou co-autor de conhecimento. Neste contexto, escolhemos utilizar a plataforma brasileira para redes sociais livres Noosfero por entender que esta satisfaz as necessidades imediatas do projeto. Além disso, queremos aproveitar nossa posição geográfica favorável para contribuir com a comunidade da mesna, uma vez que a maior parte de seus desenvolvedores estão localizados no Brasil. Pretendemos também colaborar para a implementação de um protocolo de federação para o Noosfero de forma a permitir a comunicação e a troca de conteúdo entre outras instâncias deste ou de outras redes que implementem o mesmo protocolo, tornando-se assim um nó dentro de uma rede de colaboração em potencial formada por diversas instituições de ensino, cada qual com sua própria rede.

Palavras-chaves: redes sociais. software livre. federação.

Abstract

[insert abstract... Daniel example]

This undergraduate work aims to present a theoric justification and to raise the requirements needed to enable the development of a collaborative network for the University of Brasília. This network acts as a virtual environment that allows users to create and share knowledge in a collaborative and horizontal way. This approach allows to change the "traditional" professor-student relationship and promotes students to act more as either an author or co-author of the knowledge produced. In this context, we selected a brazilian free social networking tool Noosfero. It meets the UnB Community project immediate needs. Furthermore, we want to use our geographical position to contribute to the community maintaining this plaform since it has most of its developers located in Brazil. We also intend to collaborate with the development of a federation protocol for Noosfero. This feature enables communication and exchange of content between other instances of this or other networks that implement the same protocol. Thus, it may become a node within a potential collaborative network formed by several educational institutions, each with it's own network.

Key-words: social networking. open-source. federation.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
JSON	JavaScript Object Notation
MVC	Model-View-Controller
Rails	Ruby on Rails
SSL	Secure Socket Layer
TCC	Trabalho de Conclusão de Curso
TLS	Transport Layer Security
UnB	Universidade de Brasília
USP	Universidade de São Paulo
W3C	World Wide Web Consortium

Sumário

1 Introdução

A utilização de redes sociais têm se tornado cada vez mais comum na vida das pessoas, em especial nas gerações mais jovens. Neste contexto, não estamos nos referindo apenas ao Facebook, Orkut e Twitter. (**??**) Atualmente, é bem comum empresas terem suas próprias redes sociais para promoverem a interação entre os seus funcionários e colaboradores. A ideia de não depender de redes centralizadas, como é o caso das supracitadas, faz com que as campanhas de Barack Obama [1], nos Estados Unidos, e DilmaRousseff [2], no Brasil, sejam exemplos da necessidade de autonomia das suas redes (e informações delas) e como melhor explorar a Internet, que tem em sua concepção ser descentralizada. Em especial, as tecnologias de mídia social exercem um papel importante para essa descentralização uma vez que possuem como característica a criação e o compartilhamento de conteúdo de forma horizontal e colaborativa. Baseado nessa visão, a proposta desse trabalho é disponibilizar um ambiente virtual que possibilite que os alunos da Universidade de Brasília criem e compartilhem conhecimento de forma colaborativa e horizontal, sendo assim, complementar ao modelo vertical de ensino adotado em sala de aula.

Estudos mostram que a grande maioria dos alunos de graduação faz uso de algum tipo de rede social e que aqueles que participam de forma mais ativa delas tendem a obter maior riqueza em suas relações sociais . Também, deve-se enfatizar que, está no cerne da criação das instituições de ensino superior universalizar o conhecimento. Compartilhar e dar subsídios para que o conhecimento seja disseminado e reproduzível é um dos pilares da ciência. Adaptando essas ideias para o ponto de vista tecnológico, sites de rede social são apenas uma camada das tecnologias de mídia social. A definição mais ampla de tecnologias de mídia social inclui: a totalidade de "produtos"e "serviços"digitais disponibilizados online; o comportamento social gerado pelo usuário; e a permutação de conteúdo gerado primariamente pelos próprios usuários.

As tecnologias de mídia social também afetaram as universidades, de forma que o número de perfis ou páginas oficiais de universidades é cada vez mais comum, chegando a atingir 100% das instituições de graduação americanas. As universidades passaram a utilizar essas tecnologias como forma de promover uma interação maior com seus integrantes. As tecnologias de mídia social deram às instituição de ensino a oportunidade de divulgar as conquistas de seus alunos, criando assim um sentimento de lealdade e ao mesmo tempo atraindo alunos em potencial (**??**). Além disso, foi constatado que alunos que utilizam tecnologias de mídia social com o propósito de realizar atividades acadêmicas possuem maior nível de engajamento nelas (**??**). No entanto, a maioria das redes sociais

[1] http://my.barackobama.com
[2] http://dilmanarede.com.br

utilizadas por esses alunos e universidades são as mesmas que tendem a querer centralizar as atividades nelas e , em geral não costumam fornecer mecanismos específicos para o intercâmbio de conhecimento científico e acadêmico.

Sob essa perspectiva das redes sociais no âmbito acadêmico, nós pretendemos viabilizar a implantação de uma rede social própria para a Universidade de Brasília baseada na ferramenta de software-livre Noosfero [3] na qual os usuários poderão publicar e compartilhar conteúdo livremente e colaborar com a difusão dessa nuvem de conhecimento e ideias que a Universidade nos proporciona. Pretendemos também colaborar com a comunidade do Noosfero na implementação de um protocolo de federação de forma a possibilitar no futuro a criação de uma rede de colaboração federada na qual cada universidade poderá ter seu próprio nó, de forma independente, mas ao mesmo tempo integrada aos demais.

O conceito de rede social federada é usado para indicar uma rede social autônoma controlada por uma entidade mas que possibilita a interação, através de regras acordadas, com usuários ou entidades de outras redes federadas sem a necessidade de criar uma conta na segunda.

1.1 Objetivos

Esta seção apresenta os objetivos gerais e específicos deste TCC.

1.1.1 Objetivos Gerais

Neste trabalho de conclusão de curso, temos o objetivo de implementar as principais funcionalidades para que uma rede de colaboração, como o Stoa da USP, possa ser também disponibilizada na Universidade de Brasília. Além disso, queremos aproveitar nossa "posição geográfica", com o fato do Noosfero ser uma plataforma de software livre que tem seus principais desenvolvedores no Brasil, para colaborarmos com o desenvolvimento dessa plataforma, interagindo diretamente com essa comunidade. Planejamos contribuir com a implementação e adaptação de um protocolo de federação para Noosfero, o que permitirá testarmos algumas funcionalidades da federação entre as redes acadêmicas da USP e da versão que iremos propomos como rede de colaboração da UnB.

1.1.2 Específicos

Os objetivos específicos desse trabalho são:

1. Identificar as tecnologias utilizadas pela UnB através das quais será possível autenticar-se na comunidade.

[3] http://noosfero.org/

2. Implantar uma instância do Noosfero e disponibilizar para a comunidade.
3. Integrar a instância do Noosfero às tecnologias cabíveis.
4. Adequar a instância do Noosfero ao padrão visual da UnB.
5. Levantar junto aos estudantes um conjunto de funcionalidades a serem encorporadas ao Noosfero para disponibilizar um ambiente virtual adequado ao ensino.
6. Desenvolver novas funcionalidades selecionadas.
7. Pesquisar a respeito de protocolos de federação.
8. Colaborar com a comunidade do Noosfero para a implementação do protocolo de federação escolhido.

1.2 Organização do Trabalho

Durante a primeira fase do projeto, o trabalho foi conduzido na forma de pesquisas de fundamentação teórica sobre o uso de mídias sociais na educação e sobre as tecnologias a serem utilizadas em nas fases mais avançadas. Posteriormente, nos reunimos para instalar uma instância do Noosfero em um servidor de testes disponibilizadas para nós pelo Centro de Difusão de Tecnologia e Conhecimento (CDTC).

As pesquisas para levantamento de funcionalidades junto aos estudantes da universidade será feita através de questionários com questões objetivas e subjetivas. Para a fase de desenvolvimento do trabalho, utilizaremos algumas práticas das metodologias ágeis como a realização de ciclos curtos de desenvolvimento, programação em par, sempre que possível envolvendo membros da comunidade mantenedora do Noosfero, e a realização de testes já nas primeiras fases do desenvolvimento.

Para a primeira fase deste Trabalho de Conclusão de Curso, além desta introdução este texto está organizado em capítulos. O Capítulo 2 apresenta a fundamentação teórica sobre o uso de mídias sociais na educação. O Capítulo 3 apresenta a plataforma para criação de redes sociais livros, Noosfero, e aborda a importância da adoção de software livre na educação. Por fim, o Capítulo 4 apresenta o estado atual da rede de colaboração da UnB, atualmente chamada Comunidade UnB, e as atividades planejadas para a continuidade do projeto.

2 Software Livre

Achei interessante falar um pouco sobre software livre, já que o Mezuro se encaixa nessa definição. Lembro do sr. ter falado um pouco sobre colaboração dos usuários do Mezuro com relação as métricas de determinado projeto (creio que foi isso). Não sei se essa divisão está boa, não procurei muitos trabalhos para tomar como base.

1. SOFTWARE LIVRE COMO - MÉTODOS ÁGEIS 2. MAPEAMENTO ENTRE PRÁTICAS ÁGEIS E DA COMUNIDADE SW LIVRE (PROCURAR DISSERTAÇÃO DE MESTREDO HUGO DISSERTAÇÃO PROF. PAULO)

3 Evolução de Software

Tenho dúvidas na separação desses dois conceitos (manutenção e evolução) do trabalho desenvolvido com o Mezuro. Vejo o trabalho com o Mezuro como um processo de evolução. Por outro lado, sempre vejo evolução atrelada a manutenção na literatura. Assim como o propósito da evolução do Mezuro se encaixa no propósito de uma evolução preventiva, facilitar a manutenbilidade.

Possíveis tópicos (Baseados no livro: Software Maintenance – Franz Wotowa):

Motivação Definição Classificação de Mudanças O Processo de Manutenção Leis de Lehman

4 Arquitetura de Software

O desenvolvimento de um sistema de software não é uma tarefa simples por conta da complexidade envolvida no seu processo de desenvolvimento. Além de lidar com a complexidade inerente ao problema a ser resolvido, devemos nos preocupar em como o software resolve esse mesmo problema. Por esse motivo muitos softwares fracassam, seja por custar muito acima do orçamento, estarem imcompletos ou não solucionarem os problemas como deveriam. Assim um software além de resolver o problema, deve resolvê-lo da forma esperada, satisfazendo atributos de qualidade (**??**).

Conforme a complexidade dos softwares aumentam os problemas de design vão além de algorítmos e estruturas de dados. A especificação da estrutura geral do sistema surge como um novo tipo de problema (**??**). Visando amenizar problemas como esse, a arquitetura de software tem recebido grande atenção desde a década passada, já que ela tem auxiliado na obtenção de ótimos resultados quanto ao atendimento de atributos de qualidade (**??**).

Desde a primeira referência em um relatório técnico intitulado Software Engineering Tecnhiques (**??**), na década de 1970, diversos autores buscaram definir o termo engenharia de software. Abaixo se encontram algumas das definições de alguns dos autores que se destacaram na área.

Baseados no trabalho de Mary Shaw e David Garlan (Shaw and Garlan 1996), Philippe Kruchten, Grady Booch, Kurt Bittner, e Rich Reitman construíram a seguinte definição para Arquitetura de software:

"Arquitetura de Software engloba o conjunto de decisões significativas sobre a organização de um sistema de software incluindo a seleção de elementos estruturais e suas interfaces pelos quais um sistema é composto, o comportamento como especificado em colaboração entre aqueles elementos, composição desses elementos estruturais e comportamentais dentro de um subsistema maior, além de um estilo arquitetural que orienta essa organização. Arquitetura de Software também envolve funcionalidade, usabilidade, flexibilidade, desempenho, reuso, compreensibilidade, restrições econômicas e tecnológicas, vantagens e desvantagens, além de preocupações estéticas."

De acordo com as definições acima é percebido que não há um consenso ou padrão entre os autores tanto do ponto de vista conceitual como a forma de representar uma arquitetura de software (**??**). Essa foi a principal motivação para a criação da ISO/IEEE 1471-2000. Com o intuito de estabelecer um consenso entre autores, profissionais, professores e estudantes da área sobre o que é e para que serve a arquitetura de software, esse padrão não só define o termo mas também introduz um conjunto de conceitos relacionados

a arquitetura. Esse padrão define arquitetura de software como:

"Arquitetura é a organização fundamental de um sistema incorporada em seus componentes, seus relacionamentos com o ambiente, e os princípios que conduzem seu design e evolução."

Embora exista falta de consenso, há três conceitos citados por todos os autores quando se trata de arquitetura de software (**??**). São eles:

- Elementos estruturais ou de software, também chamados de módulos ou componentes, são as abstrações responsáveis por representar as entidades que implementam funcionalidades especificadas.
- Interfaces ou relacionamentos, também chamados de conectores, são as abstrações responsáveis por representar as entidades que facilitam a comunicação entre os elementos de software.
- Organização ou configuração que consiste na forma como os elementos de software e conectores estão organizados.

Analisando o ciclo de desenvolvimento de software, é observado que a arquitetura é uma abordagem empregada desde as fases iniciais do processo. Neste instante o nível de abstração da arquitetura é bastante elevado, tendo o objetivo de definir e apresentar a solução computacional que será implementada, auxiliando a tomada de decisões dos stakeholders ou envolvidos no processo de desenvolvimento. Contudo, ao longo do desenvolvimento do software, a arquitetura sofre refinamentos que diminuem o nível de abstração e permitem, por exemplo, a representação dos relacionamentos entre os elementos arquiteturais e os arquivos de código fonte responsáveis por implementá-los (**??**).

Para que os stakeholders possam visualizar e utilizar a arquitetura como entrada para demais atividades do processo, ou para a tomada de decisões, é necessário que ela esteja representada, ou documentada, de forma que seja possível utiliza-la para esses fins. A essa representação é dado o nome de documento arquitetural, o qual é composto por um conjunto de modelos e informações que descrevem principalmente a estrutura do software especificado para atender aos requisitos (**??**).

Qualquer software possui arquitetura, independente dela ser documentada ou projetada. Porém tendo apenas uma arquitetura implementada, ou seja, arquitetura sem projeto, deixa-se de obter os benefícios ou vantagens que uma arquitetura projetada e bem documentada pode oferecer. Entre os benefícios da documentação da arquitetura estão a possibiblidade dela se tornar uma ferramenta de comunicação entre os stakeholders do projeto, um método ou modelo para a análise antecipada do sistema a ser desenvolvido, assim como uma ferramenta que permite a rastreabilidade entre os requisitos e os elementos que compõem o sistema.

Como ferramenta de comunicação, a arquitetura de um software deve servir aos principais envolvidos, já que para cada um, a arquitetura do software é utilizada com diferentes propósitos (**??????**):

- Cliente. O cliente é a pessoa ou empresa que contrata uma equipe de desenvolvimento para a construção de um sistema de sua necessidade. Na fase inicial do projeto, ele necessita de uma estimativa de certos fatores, normalmente econômicos, que podem ser obtidos após a definição da estrutura principal do software. O cliente, por exemplo, tem interesse em estimativas de custo, confiabilidade e manutenibilidade do software que podem ser obtidos principalmente através de uma análise da arquitetura. Portanto, é de extrema importância para o cliente que a arquitetura atenda os requisitos do software de forma a representar suas reais expectativas em relação ao que foi especificado.

- Gerentes. A arquitetura permite aos gerentes tomarem certas decisões de projeto por possibilitar a sumarização das diversas características do sistema. Um gerente pode, por exemplo, usar a arquitetura como base para definir as equipes de desenvolvimento de acordo com os elementos arquiteturais que estão identificados e que devem ser construídos.

- Desenvolvedor. Da arquitetura de um software, o desenvolvedor busca uma especificação que escreva a solução com detalhes suficientes e que satisfaça os requisitos do cliente, mas que não seja tão restritiva a ponto de limitar a escolha das abordagens para a sua implementação. Os desenvolvedores usam a arquitetura como uma referência para a composição e o desenvolvimento dos elementos do sistema, e para a identificação e reutilização de elementos arquiteturais já construídos.

- Testadores. A arquitetura fornece, numa visão de caixa preta, informações aos testadores relacionadas ao correto comportamento dos elementos arquiteturais que se integram. Sendo assim, este artefato pode ser um dos artefatos base utilizados durante o planejamento e execução de testes de integração e de sistema.

- Mantenedor. A descrição arquitetural do software fornece aos mantenedores uma estrutura central da aplicação que idealmente não deve ser violada. Qualquer mudança deve preservá-la, buscando, se possível, uma modificação puramente dos elementos arquiteturais e não da forma como estão organizados.

[Porque utilizar multiplas visões? Utilizar livro do Guilherme Germoglio como referência]

[Descrever Atributos de Qualidade, e como são alcançados. –> Link para decisões arquiteturais]

4.1 O Framework Ruby on Rails

O Ruby on Rails é um framework open-source criado em 2003 por David Heinemeier Hansson, extraído de um software, um gerenciador de projetos conhecido como Basecamp. Hansson lançou o Rails pela primeira vez ao público em 2004, e desde então seu desenvolvimento e utilização são cada vez maiores. Diversos programadores e empresas em todo mundo utilizam esse o Rails para construirem suas aplicações. Entre as mais conhecidas estão o Twitter, GitHub e Groupon.

O Rails utiliza a linguagem de programação Ruby, criada no Japão em 1995 por Yukihiro "Matz"Matsumoto. A linguagem Ruby é interpretada, multiparadigma, com tipagem dinâmica e gerenciamento de memória automático. O Rails é basicamente uma biblioteca Ruby ou *gem* e é construído utilizando o padrão arquitetural MVC.

4.1.1 Padrao Arquitetural MVC

O padrão MVC começou como um framework desenvolvido por Trygve Reenskaug para a plataforma SmallTalk no final dos anos 70. Desde então ele exerce grande influencia sobre diversos frameworks que promovem interação com usuário.

O MVC visa separar a representação da informação da interação com o usuário. Para atingir esse objetivo são utilizados três papéis. O modelo (model) que representa informações do domínio, como dados da aplicação, regras de negócio, lógica e funções. A visão (view) que são saídas de representação dos dados do modelo ao usuário. Um exemplo comum de visão é uma pagína HTML contendo dados presentes no modelo. O útimo papel, o controlador (controller) é responsável por receber requisições da visão, manipula-las, utilizando dados do modelo, e atualizar a visão para satisfazer as requisições do usuário.

[Inserir figura que represente esse padrão]

[Inserir parágrafo exemplificando o funcionamento do padrão - requisição http simples com um browser qualquer]

4.1.2 Arquiterura do Rails

[Inserir elementos estáticos e dinâmicos presentes no Rails]

4.1.3 Evolução do Ruby on Rails

4.1.4 Plugins no Ruby on Rails

5 Mezuro: Uma plataforma de Monitoramento de Código Fonte

A prática da engenharia de software exige compreensão do projeto como um todo, onde o código-fonte é uma das partes mais importantes. O engenheiro de software precisa analizar um código-fonte diversas vezes, seja para desenvolver novas funcionalidades ou melhorar as existentes (**??**).

Como parte fundamental do projeto de software, o código-fonte é um dos principais artefatos para avaliar sua qualidade (**??**). Essas avaliações não são meramente subjetivas, sendo necessário extrair informações que possam ser replicadas e entendidas da mesma forma, independente de quem analisa o código. As métricas de código-fonte permitem esse tipo de avaliação pois possibilitam analisar, de forma objetiva, as principais características para aceitação de um software.

Há várias características que fazem do software um sistema de qualidade ou não. Entre elas há algumas que são obtidas exclusivamente através do código-fonte. Quando compilamos um software, por exemplo, características podem ser analisadas, mas outras como organização e legibilidade não. Isso não refletiria tamanho, modularidade, manutenbilidade, complexidade, flexibilidade, que são características encontradas na análise de códigos-fonte (**??**).

Extrair métricas de um código-fonte manualmente, é uma tarefa bastante difícil, independente do tamanho do código. Porém quanto maior e mais complexo é o código, essa tarefa se torna praticamente impossível pela quantidade de atributos e linhas de códigos a se analisar, além de não ser viável pelo tempo despendido nessa tarefa. Por outro lado há ferramentas CASE para monitoramento automático de códigos-fonte. Para isso existem poucas ferramentas disponíveis no mercado, e muitas delas nem sempre são adequadas para análise do projeto alvo (software-livre) (**??**).

A partir da necessidade de extrair métricas de código-fonte e interpretar seus valores possibilitando o monitoramento de características específicas do software foi desenvolvida uma plataforma chamada Mezuro[1]. O Mezuro porém não foi criado da noite para o dia. Ele foi concebido através de um longo processo de amadurecimento de diversas ferramentas, que teve seu início com o projeto Qualipso[2].

[1] http://mezuro.org/
[2] http://qualipso.icmc.usp.br/

5.1 Concepção do Projeto Mezuro

O projeto (Qualipso Quality Platform for Open Source) é hoje um consórcio formado por indústria, academia, e governo. Seu principal objetivo é potencializar as práticas de desenvolvimento de software livre, tornando-as confiáveis, reconhecidas e estabelecidas na indústria, através da implementação de tecnologias, procedimentos e leis (**??**).

Uma das iniciativas desse projeto envolvia a adaptaçao da ferramenta JaBUTi - Java Bytecode Understanding and Testing, uma ferramenta de análise de softwares em linguagem Java. O objetivo inicial era melhorar o módulo de cálculo das métricas fornecido pela JaBUTi. Porém a proposta foi alterada para a visualização de métricas, com configuração de intervalos das mesmas, facilitando a análise dos resultados. Esse módulo de visualização, posteriormente, se transformou em uma aplicação independente, o Crab (**??**). Neste instante o JaBUTi calculava asa métricas então chamava o Crab para visualiza-las, assim como possibilitar configurações e criação de métricas compostas. Após a experiência com a JaBUTi como ferramenta base, ou seja, aquela que coleta métricas de código-fonte, buscou-se outras alternativas para essa tarefa já que a JaBUTi era restrita a códigos escritos em liguagem Java, o que limita a contribuição com softwares livre.

Uma alternativa encontrada foi o software livre denominado Analizo[3], o qual surgiu a partir de um outro software livre denominado Egypt[4]. Após diversas contribuições (principalemnte de Antônio Terceiro e do grupo CCSL-USP) resultou em diversas funcionalidades que fugiram do escopo inicial do Egypt, que passou a ser o Analizo ("análise"em esperanto).

Na integração do Analizo com a Crab, este último consome serviços do Analizo, ao contrário do que acontecia na integração da JaBUTi com a Crab, onde a ferramenta base aciona o módulo de visualização (Crab) após a coleta das métricas. Essa inversão de paradigma e outras mudanças de impacto fez com que a Crab fosse relicenciada como LGPL versão 3, e para padronizar o nome das ferramentas a Crab passou a se chamar Kalibro ("calibrar"em esperanto). Em sua primeira versão para o projeto Qualipso o Kalibro possuia base de dados própria, calculava estatísticas para as métricas coletadas e exibia os resultados baseando-se nas configurações cadastradas.

Após a primeira versão do Kalibro integrada ao Analizo, decidiu-se separar a coleta, configuração e interpretação de métricas da sua visualização. Com isso surgiu o Kalibro Service, um serviço web sem interface gráfica, o qual possui todas as funcionalidades não-visuais que hoje são fornecidas pela plataforma Mezuro (**??**).

[3] http://analizo.org/
[4] http://gson.org/egypt/

5.2 Mezuro como Plugin do Noosfero

O Mezuro foi concebido como instância de uma plataforma web conhecida como Noosfero com o plugin Mezuro ativado. O Noosfero é um software livre para criação de redes sociais desenvolvido pela Cooperativa de Tecnologias Livres - Coolivre sob licença AGPL v.3 com o intuito de permitir com que os usuários criem sua própria rede social livre e personalizada de acordo com suas necessidades.

A linguagem de programação Ruby versão 1.8.7 e o framework MVC Ruby on Rails foram utilizados para desenvolver o Noosfero. Essas tecnologias foram escolhidas pois a linguagem Ruby possui uma sintaxe simples, que facilita a manunbilidade do sistema, característica importante em projetos de software livre que tendem a atrair colaboradores externos a equipe. Já o framework Ruby on Rails influencia em maior produtividade graças a conceitos como *convention over configuration* e DRY [baseado TCC Daniel].

A arquitetura do Noosfero permite a adição de novas funcionalidades através de plugins. Essa característica é interessante pois colaboradores podem implementar novas funcionalidades sem entender ou interagir com o código-fonte do Noosfero, já que os plugins possuem o código isolado, mantendo o baixo acoplamento e alta coesão dos módulos do sistema. Embora plugins sejam totalmente independentes do sistema alvo, no Noosfero os plugins são mantidos com o código principal para auxiliar no controle de qualidade do ambiente. Pensando nisso os plugins devem ter testes automatizados para quando houver a necessidade de alterar o código do Noosfero, os testes dos plugins devem ser executados para verificar se as mudanças não afetaram seu funcionamento (**??**).

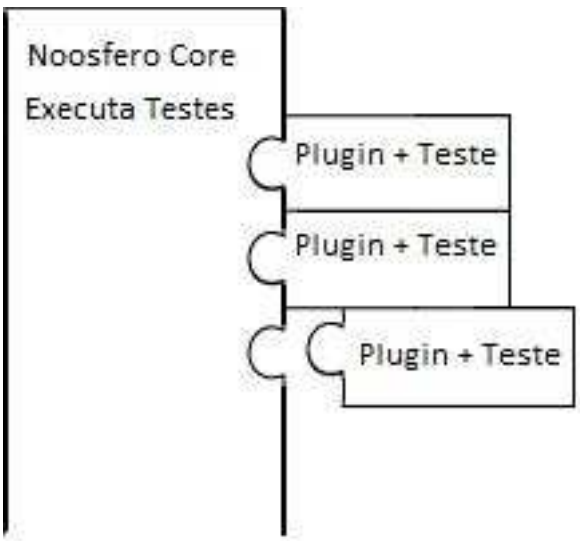

Figura 1 – Arquitetura de Plugins do Noosfero

O Mezuro como plugin do Noosfero foi construído utilizando o recurso de plugins do framework Ruby on Rails. Por esse motivo ele herda a arquitetura desse framework,

a qual é baseada no padrão arquitetural MVC, assim como o Noosfero. O Mezuro utiliza um web service chamado Kalibro Metrics. Isso permite ao Mezuro:

- Baixa códigos-fonte de repositórios dos tipos GIT, Subversion, Baazar e CVS
- Criação de configuraçções as quais são conjuntos pré-definidos de métricas relacionadas para serem utilizadas na avaliação de projetos de software.
- Criação de intervalos relacionados com a métricas e avaliações qualitativas.
- Criação de novas métricas de acordo com aquelas fornecidas pelos coletores do Kalibro.
- Cálculo de resultados estatísticos para módulos com alta granularidade.
- Possibilidade de exportar arquivos com os resultados gerados.
- Interpretação dos resultados com inteface mais amigável aos usuarios com a utilização de cores nos intervalos das métricas.

5.3 Mezuro como aplicação independente

O framework Rails, desde seu lançamento, sofre frequentes alterações, e com isso novas versões são lançadas constantemente. Muitos desenvolvedores enxergam essas constantes mudanças como um ponto negativo, ao passo que há incompatibilidade de algumas "gems"de uma versão para outra. Por outro lado, outros aprovam essa característica pois a cada atualização há melhora no produto, com adição de novos recursos, além de ser um incentivo para a implementação de testes, já que eles auxiliam a manter a integridade da aplicação a cada atualização.

Acompanhando a evolução do framework, a equipe de desenvolvimento da plataforma Mezuro decidiu atualiza-la para as novas versões oferecidas do Ruby on Rails e da linguagem Ruby, as quais possuem atualmente as versões 4 e 2, respectivamente. Um dos motivos para essa atualização é aproveitar os novos recursos oferecidos, além do suporte da comunidade[5], que favorece a utilização das versões mais novas.

Entre os novos recursos oferecidos pela versão 4 do rails estão:

- Páginas mais rápidas através da utilização de Turbolinks. Ao invés de deixar o navegador recompilar o JavaScript e CSS entre cada mudança de página, a instância da página atual é mantida, substituindo apenas o conteúdo e o título.

[5] http://rubyonrails.org/

- Suporte para a expiração de cache baseado em chave, que automatiza a invalidação do cache e deixa mais fácil a implementação de estruturas de cache sofisticadas.
- Streaming de vídeo ao vivo em conexões persistentes.
- Melhorias no ActiveRecord para aprimorar a consistência do escopo e da estrutura das queries.
- Padrões de segurança locked-down.
- Threads seguras por padrão e a eliminação da necessidade de configurar servidores com thread.