Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática

Ferramentas de Suporte à Gestão

projecto realizado na

Nokia Siemens Networks

por

Tiago José Pombas Santiago Marques Honorato

Mestrado em Engenharia Informática

2007

Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática

Ferramentas de Suporte à Gestão

projecto realizado na

Nokia Siemens Networks

por

Tiago José Pombas Santiago Marques Honorato

Projecto orientado pelo Professor Doutor Pedro Antunes
e co-orientado pelo Engenheiro Frederico Figueiredo

Mestrado em Engenharia Informática

2007

Declaração

Tiago José Pombas Santiago Marques Honorato, aluno nº 29157 da Faculdade de Ciências da Universidade de Lisboa, declara ceder os seus direitos de cópia sobre o seu Relatório de Projecto em Engenharia Informática, intitulado "Ferramentas de Suporte à Gestão", realizado no ano lectivo de 2006/2007 à Faculdade de Ciências da Universidade de Lisboa para o efeito de arquivo e consulta nas suas bibliotecas e publicação do mesmo em formato electrónico na Internet.

FCUL, 26 de Junho de 2007

Frederico Figueiredo, supervisor do projecto de *Tiago José Pombas Santiago Marques Honorato*, aluno da Faculdade de Ciências da Universidade de Lisboa, declara concordar com a divulgação do Relatório do Projecto em Engenharia Informática, intitulado "Ferramentas de Suporte à Gestão".

Lisboa, 26 de Junho de 2007

Resumo

No presente, a gestão de informação enfrenta vários desafios, sendo cada vez mais complexos em empresas de grande dimensão , como a Nokia Siemens Networks (NSN). Para diminuir essa complexidade, este trabalho analisa alternativas de protótipos para suportar a gestão de conteúdos e projectos no ambiente NSN.

Para suportar a gestão de actividades e conteúdos, para uma equipa específica da NSN, começou-se por identificar todos os requisitos necessários para cumprir todas as funcionalidades pretendidas. De seguida, vários *Content Management System* (CMS) *open source* foram analisados, procurando por uma ferramenta base pra integrar no processo de trabalho da equipa. Adicionalmente, foi produzido um protótipo de baixa-funcionalidade e um novo conceito para simplificar, editar, e partilhar informação, foi desenvolvido.

Mais tarde, devido ao vasto número de projectos realizados na NSN, as equipas de gestão reconheceram a necessidade de estender as funcionalidades das suas ferramentas correntes, para facilitar a tarefa de gestão de projectos e recursos. Como resultado, as ferramentas actuais foram estudadas e extensões para permitir uma visualização gráfica, não só da duração de tarefas, associadas a um projecto, mas também dos recursos ligados aos projectos (para valores actuais e planeados), foram desenvolvidas. Com esta nova funcionalidade, os gestores de projectos são agora capazes de visualizar melhor, e de uma forma simples, o estado do projecto corrente.

PALAVRAS-CHAVE:
Gestão de Informação, Content Management System, What You See Is What You Get, Web Services

Abstract

Nowadays, information management faces diverse and well-known challenges, which are even more complex in large companies, such as Nokia Siemens Networks (NSN). To address this issue, this work analyzes and prototypes alternatives to support the management of contents and projects, in the NSN environment.

To support contents and activities management, for a specific team at NSN, the first thing to do was identify all the requirements needed to fully address their needs. Then, various open source Content Management Systems (CMS) were surveyed, seeking for a base tool to integrate into the team's work process. In addition, low fidelity prototypes of the user interface and a new concept for simplifying the process of creating, editing, and sharing information, were produced and developed.

Later on, due to the vast number of projects taking place at NSN, management teams recognized the need to extend the functionality provided by their current tools, to facilitate the demanding task of managing projects and resources. As a result, the currently available tools were studied and extensions to allow graphic visualization, not only of the tasks' duration, associated to a project, but also of the resources linked to projects (for both actual and planned values), were developed. With this new functionality, project managers are now able to better visualize, in a simple and valuable manner, the project(s) current status.

KEYWORDS:
Information Management, Content Management System, What You See Is What You Get, Web Services

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Hoje em dia, com a acumulação de informação nas organizações, há que saber geri-la. Esta gestão corresponde à análise da informação das organizações transformando-a em acções conducentes à tomada de decisão, tendo como objectivos garantir que a informação é gerida como um recurso indispensável, e valioso, e garantir que está alinhada com os objectivos de negócio. Logo, a gestão de informação não é um fim em si mesma, mas um suporte indispensável para um objectivo mais alargado: a gestão de uma organização [1]. Neste contexto dois projectos foram realizados: Ferramentas de Suporte à Gestão de Conteúdos e Suporte à Gestão de Projectos.

1.1 Motivação

Devido à grande quantidade de projectos em inúmeras áreas, desenvolvidos pela Nokia Siemens Networks (NSN)[1], alguns colaboradores têm de se associar a diversas tarefas simultaneamente. Estes colaboradores necessitam de criar conteúdos online com vista a partilhar, por exemplo, os seus resultados com os seus membros de equipa ou com outras equipas. Até ao momento, ainda não foi adoptada uma ferramenta que suporte este processo de forma simples, rápida e com baixo custo de manutenção.

Adicionalmente, a NSN usa uma ferramenta de *Business Process Re-engineering* (BPR) (análise e *redesign* de *workflow* dentro e entre as empresas) que permite monitorizar, apoiar e acompanhar tarefas de determinados projectos na área de *Network Management*. Por estas razões, a ferramenta é de grande utilidade para a gestão estratégica de equipas e projectos. No entanto, actualmente, este produto não permite visualizar como decorre certo projecto, no que diz respeito às tarefas associadas estarem dentro do prazo ou não, nem quantos recursos estará a gastar.

Devido ao aumento da informação dentro de uma empresa como a NSN , o conceito de *Content Management System* CMS poderia ser adoptado. Com este conceito, criar, modificar, arquivar e remover informação de recursos é possível de

[1] Acessível a partir de http://www.nokiasiemensnetworks.com/

uma forma simples, sem custos elevados de aprendizagem. Também com o aumento da utilização da *World Wide Web* por parte de aplicações, com vista a comunicarem entre si, os *Web Services* aparecem como uma solução viável [2]. Estes asseguram compatibilidade não só entre aplicações novas e outras já existentes, como também, entre aplicações ou ferramentas desenvolvidas em tecnologias diferentes.

1.2 Objectivos

Para suportar as equipas que necessitam de gerir conteúdos, é necessário fazer o levantamento de requisitos, junto dos responsáveis do projecto, para que se identifique quais os requisitos mandatórios ou opcionais. Com base nos resultados obtidos, e de forma a minimizar os custos, uma proposta de adaptação ou elaboração de um portal *web* parece apropriado, depois de identificada a ferramenta que suporte todos ou a maioria dos requisitos. Com vista a permitir a utilizadores criar/editar conteúdos sem possuírem conhecimentos técnicos para tal, um protótipo será construído.

Actualmente, visto que os gestores de projectos não conseguem controlar e monitorizar os recursos gastos em determinados projectos a partir da ferramenta existente, então, novas funcionalidades deverão ser implementadas. Para ser possível a utilização de *Web Services* como o método mais adequado para estender as funcionalidades actuais, um novo ambiente terá de ser configurado, uma vez que até à data, a ferramenta recolhe os dados de determinado projecto(s) directamente da base de dados (gerida por uma ferramenta de monitorização). Os *Web Services* serão responsáveis pela gestão de recursos, projectos ou portfolios e imputação de horas, que permitem aceder à base de dados pretendida e recolher ou actualizar esses mesmos dados. Com estes dados recolhidos, e devidamente armazenados, será possível atingir o objectivo final - representar esses dados graficamente, de modo a facilitar a análise dos mesmos por parte do gestor do projecto.

1.3 Organização do documento

Este documento está dividido em quatro capítulos. No Capítulo 2 encontra-se o enquadramento organizacional que aborda a transição da empresa Siemens para a empresa NSN, bem como o funcionamento da equipa onde se insere o projecto. Posteriormente, no Capítulo 3, são descritos ao pormenor os projectos realizados, explicando a contextualização, as tecnologias e ferramentas utilizadas para desenvolver cada projecto, o trabalho realizado, os resultados adquiridos em cada um e os resultados globais do estágio. Finalmente, as conclusões e o trabalho futuro são apresentadas no capítulo 4.

Capítulo 2

Enquadramento Organizacional

Neste capítulo irá ser apresentada, sucintamente, a história da organização desde a sua fundação até aos dias de hoje, e a área onde o projecto foi inserido. Esta breve apresentação tem como objectivo a percepção, por parte do leitor, sobre o ambiente social e profissional em que o projecto foi desenvolvido.

2.1 Siemens

Em 1847 foi fundada a Siemens AG por Werner von Siemens com o intuito de instalar linhas telegráficas e fabricar o telégrafo de ponteiro. Este produto foi desenvolvido pelo próprio Werner e por Johann Georg Halske, diferente do telégrafo comum (usava código morse), tendo uma tecla para cada letra do alfabeto, permitindo a qualquer adulto conhecedor do alfabeto utilizá-lo. Desde então, a empresa foi crescendo e, em 1870, instalou a linha telegráfica Indo-Europeia, permitindo ligar Londres a Calcutá - um dos trabalhos mais famosos da Siemens. Onze anos depois, em 1881, a Siemens instalou a primeira rede de iluminação eléctrica de rua da Europa. A empresa foi crescendo e foi-se dividindo nas mais diversas áreas de negócio, tornando-se mesmo na maior empresa de electrónica do mundo.

No presente, a Siemens é uma empresa multinacional, com 500 centros de produção em 50 países, com aproximadamente 500 mil colaboradores. A Siemens está representada em todo o mundo nas mais diversas áreas.

Este projecto foi inserido no *Operating Group* (OG) *Information and Communications* (IC), um dos principais participantes na indústria das telecomunicações. A Siemens IC é líder mundial na inovação de produtos e serviços para redes móveis e fixas e presta os seguintes serviços: soluções para operadores de rede móvel e fixa; soluções de comunicações para empresas; ferramentas de gestão e performance de redes de telecomunicações; projectos integrados de telecomunicações; competências em investigação e desenvolvimento.

Adicionalmente, a Siemens IC oferece os seguintes produtos: equipamentos de

análise e medida profissional; integração de aplicações, network consulting, etc.; equipamentos residenciais; sistemas de comunicação e infra-estruturas de apoio, sistemas de controlo e supervisão, etc.; equipamentos Pessoais; Módulos sem fios.

Mais recentemente, no dia 2 de Abril de 2007, a Siemens fundiu-se oficialmente com a Nokia, dando início a uma nova empresa.

2.2 Nokia Siemens Networks

Sendo a Siemens uma das maiores empresas no campo da indústria das comunicações, desde que foi fundada, e sendo a Nokia líder no desenvolvimento de comunicações móveis, a fusão entre estas duas empresas originou um gigante no mundo das comunicações - a NSN.

Com a junção das empresas Siemens e Nokia, conjugou-se o melhor das duas e a NSN tornou-se numa das maiores empresas (entre as três melhores) em termos de telecomunicações: número 2 em *Wireless Networks*, número 2 em *Operator Services* e número 3 em *Wireline Networks*.

Tornou-se então possível fornecer um vasto leque de soluções para estruturas de rede fixa e móvel, dado que a NSN está presente em 150 países em sete regiões: América do Norte; América Latina; Oeste e Sul da Europa; Médio Oriente; Africa; China e APAC (*Asian Pacific*).

Os principais centros de produção encontram-se na China, Finlândia, Alemanha, India e Itália, estando o "quartel-general"localizado na Finlândia. No total, a NSN tem aproximadamente 60.000 colaboradores, que permite oferecer soluções a 600 clientes, encontrando-se, entre estes, 75% das operadoras de topo em todo o mundo.

Com esta fusão, as unidades de negócio também sofreram alterações, e, consequentemente, a área onde o projecto se inseria mudou. Neste momento, as unidades de negócio são as seguintes:

- *Radio Access*
- *Broadband Access*
- *Service Core and Applications*
- *IP/Transport*
- *Operations and Business Software*

Este projecto, na NSN, foi inserido na unidade de negócio *Operations and Business Software*, responsável por fornecer software que funcione de forma automatizada de modo a reduzir a complexidade para os operadores e, ao mesmo tempo, aumentar a performance do negócio.

2.3 User eXperience Group

A equipa onde o projecto foi integrado tem o nome de *User eXperience Group* (UXG). É uma equipa centrada na área da usabilidade, que participa em vários projectos de diversas áreas (Aplicações Internet, Desktop, Home Entertainment, etc.). O grupo assiste outras equipas de desenvolvimento na realização de testes, no design e execução de *User Interface* (UI). O UXG é uma equipa jovem, com muita motivação, com ideias novas e com grande espírito de entreajuda.

2.3.1 Funcionamento

Um mês antes do começo do estágio, em Agosto de 2006, a Siemens S.A. criou uma equipa (UXG) com o intuito de levar a cabo o desenvolvimento de projectos na área da usabilidade. Esta equipa é, actualmente, constituída por cinco pessoas, onde cada uma desempenha determinadas tarefas num certo projecto. Essas tarefas são atribuídas e previamente definidas pelo responsável da equipa. Dado que o UXG trabalha como uma equipa de consultoria, por vezes é necessário mudar de tarefas consoante a prioridade e as necessidades de um projecto. Quando tal acontece, o responsável pela equipa tem de ser notificado, para verificar, conjuntamente com os membros da equipa, quais as vantagens e desvantagens e se essa mudança de tarefa é adequada.

O responsável tem como funções reportar o trabalho desenvolvido à chefia; interagir com os clientes; entrosamento, boa interacção e organização da equipa; planeamento das tarefas a atribuir a cada elemento da equipa; estabelecimento de objectivos e de prazos para acabar os projectos.

Com o decorrer do estágio, houve a necessidade de interagir mais frequentemente com outra equipa de desenvolvimento, de *Project Management*, onde estava inserido um segundo projecto a desenvolver - Suporte à Gestão de Projectos. O UXG é um bom candidato para este projecto porque o objectivo primordial desta equipa é facilitar a vida do utilizador, permitindo que este atinja os seus objectivos com o mínimo esforço possível.

2.3.2 Integração

Após terminar a licenciatura na Faculdade de Ciências da Universidade de Lisboa e ingressar numa empresa como a Siemens S.A. (no presente NSN), houve um período de adaptação, tanto ao ambiente como ao local de trabalho. É uma adaptação a uma nova realidade, um novo ambiente, novas pessoas, uma maior responsabilidade no trabalho a desenvolver e uma rotina diária diferente da que se vivia nos tempos académicos. O início de uma nova etapa.

A integração na empresa e na própria equipa, por parte do estagiário, decorreu com toda a naturalidade. A facilidade de integração deveu-se essencialmente ao bom ambiente encontrado, principalmente, dentro da equipa em que o estagiário ingressou. O bom ambiente de trabalho e o bom relacionamento com os membros da equipa, fez com que as oito horas diárias de trabalho permitissem desenvolver um trabalho produtivo, reunindo assim todas as condições para a realização, com sucesso, dos projectos propostos.

2.3.3 Planeamento, Organização e Acompanhamento

Durante o estágio vários desafios foram encontrados, todos eles encarados com responsabilidade e profissionalismo. À medida que estes desafios iam surgindo, o responsável pela equipa era sempre avisado para que se mantivesse ao corrente das dificuldades ou do que seria preciso para ultrapassar o mesmo. Desafios esses que se baseavam, cada um, em tecnologias diferentes ou em novos conceitos. Neste cenário, e para realizar os desafios propostos com sucesso, teve de existir um trabalho de investigação antes de se passar ao próximo passo. Esta aprendizagem contínua de diferentes tecnologias e conceitos, para realizar cada tarefa do projecto final com sucesso, foi sempre acompanhada não só pelo responsável pela equipa, mas também pelos outros membros. Através de discussões com o responsável e, por vezes também, com toda a equipa, chegavam-se a conclusões indispensáveis para a realização do desafio encontrado.

Todo o trabalho desenvolvido durante uma semana de trabalho é discutido numa reunião, *Follow Up Meeting* (FUM), no final da semana, onde cada membro da equipa tem a oportunidade de apresentar aos restantes a evolução do seu trabalho e dúvidas que tenha sobre determinada tarefa, sendo mais uma ajuda para que todos os membros acompanhem todas as actividades desenvolvidas dentro da equipa. Todas as actividades realizadas durante o estágio, no primeiro projecto, estão ilustradas no mapa de Gantt na figura 2.1, enquanto que o planeamento do segundo projecto está representado na figura 2.2.

ID	Task Name	Start	Finish	Duration
1	Ramp Up	11-09-2006	29-09-2006	15d
2	Levantamento de Requisitos	02-10-2006	20-10-2006	15d
3	Análise de Gestores de Conteúdos	23-10-2006	27-11-2006	26d
4	Guidelines Genéricas de GUI para web	28-11-2006	05-12-2006	6d
5	Protótipo de Baixa-Funcionalidade	06-12-2006	10-01-2007	26d
6	Análise de editores WYSIWYG	11-01-2007	23-02-2007	32d
7	Análise de Motores de Busca	26-02-2007	09-03-2007	10d
8	Padrões de Desenho	12-03-2007	16-03-2007	5d
9	Análise de Aplicações para Testes de Regressão	19-03-2007	03-04-2007	12d

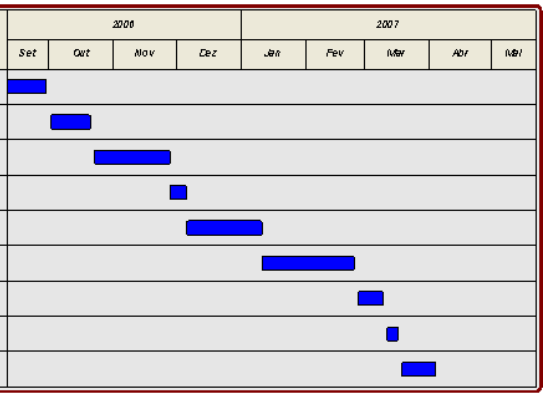


Figura 2.1: Planeamento Ferramentas de Suporte à Gestão de Conteúdos

ID	Task Name	Start	Finish	Duration
1	Ramp Up Web Services	04-04-2007	11-04-2007	6d
2	Restaurar backup da base de dados	12-04-2007	17-04-2007	4d
3	Migração da base de dados	18-04-2007	27-04-2007	8d
4	Enterprise Java Beans	02-05-2007	09-05-2007	6d
5	RPM Web Services	10-05-2007	25-05-2007	12d
6	Recolha e apresentação de dados	28-05-2007	25-06-2007	21d

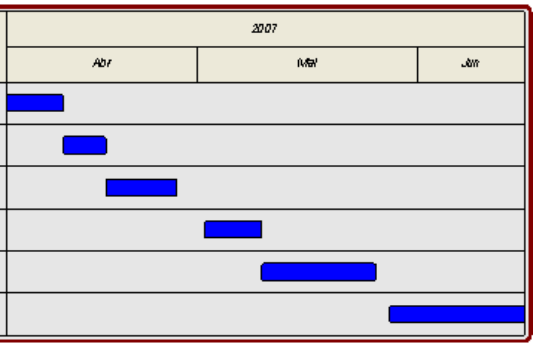


Figura 2.2: Planeamento Suporte à Gestão de Projectos

Capítulo 3

Projectos

Dois projectos foram realizados neste estágio:

- Ferramentas de Suporte à Gestão de Conteúdos;
- Suporte à Gestão de Projectos.

O projecto "Ferramentas de Suporte à Gestão de Conteúdos"consiste na identificação de um portal *Web*, que permita aos utilizadores criar ou editar conteúdos de uma forma simples, sem a necessidade de possuir conhecimentos em linguagens de programação. Após algumas reuniões com os responsáveis, os requisitos foram levantados (mandatórios e opcionais). Muitos requisitos obrigatórios foram realizados (explicados no decorrer da secção deste projecto) até que, a determinada altura e por razões alheias à equipa, o projecto não teve continuidade. O projecto parou devido ao planeamento de mudança da empresa Siemens S.A. para NSN.

A realização do projecto não estava assegurada e, à medida que o tempo passava, havia uma incerteza se a continuidade do desenvolvimento do mesmo, passaria pela Alemanha, se continuaria em Portugal ou se terminaria de vez. Perante esta incerteza, e para não prejudicar o estágio, foi proposto outro projecto - Suporte à Gestão de Projectos. O objectivo projecto seria apresentar graficamente dados recolhidos, via *Web Services*, de uma base de dados da NSN. O intuito seria facilitar a visualização do decorrer de tarefas, associadas a um determinado projecto.

De seguida serão apresentados, detalhadamente, os dois projectos.

3.1 Ferramentas de Suporte à Gestão de Conteúdos

De seguida será apresentado, em detalhe, o projecto "Ferramentas de Suporte à Gestão de Conteúdos", bem como o contexto onde se insere.

3.1.1 Contextualização

A NSN, como empresa multi-nacional, realiza projectos em diversas áreas: telecomunicações, *home entertainment*, gestão de redes, sistemas informáticos industriais, entre outras. A NSN conta com milhares de colaboradores com várias competências, muitos dos quais trabalham em múltiplas tarefas simultaneamente. Vários colaboradores, em áreas específicas (marketing, qualidade, etc.), necessitam de criar e editar conteúdos online, de forma a partilhar os seus resultados com outras ou dentro da própria equipa. O principal objectivo deste projecto foi de identificar e propor uma ferramenta, orientada às necessidades dos colaboradores com vista a reduzir o tempo e complexidade da gestão de conteúdos.

O projecto consistiu na construção e adaptação de um portal Web, onde diferentes utilizadores podem criar e editar conteúdos. Para criar/editar conteúdos os utilizadores não devem necessitar de ter conhecimentos técnicos, sendo preciso adaptar um CMS que disponibilize um editor WYSIWYG, aos requisitos dos utilizadores.

3.1.2 Tecnologias e Ferramentas

Para a realização do projecto, algumas ferramentas e tecnologias foram utilizadas. Umas mais familiares que outras, de maior ou menor dificuldade de implementação, mas todas de grande utilidade para a execução do projecto ser bem sucedida. Esta secção tem como objectivo dar a conhecer, sucintamente, quais as tecnologias e ferramentas utilizadas:

- *Hyper Text Markup Language* (HTML) [3] - linguagem utilizada para criar páginas *Web*. Fornece meios para estruturar texto num documento, usando *tags* específicas da linguagem;
- Java Script [4] - linguagem de programação frequentemente utilizada no lado do cliente usada para desenvolvimento *Web*. Esta linguagem pode ser usada para permitir o acesso a objectos integrados noutras aplicações, bem como em páginas *Web*;
- Cascade Style Sheet (CSS) [5] - linguagem que permite definir o estilo e a apresentação de uma determinada página *Web*, podendo ser aplicada também a documentos XML. A principal funcionalidade desta linguagem é separar a formatação (ou apresentação) do documento, do seu conteúdo;
- Visual Basic [6] - linguagem de programação da *Microsoft* orientada por eventos, possuindo também um ambiente gráfico de desenvolvimento que permite a construção de *Graphical User Interfaces* (GUI);

- Java [7] - linguagem de programação de alto nível orientada a objectos. A plataforma Java Enterprise Edition 5 (JEE5) foi escolhida para o desenvolvimento porque nesta versão há um aumento de desempenho, suportando tecnologias de ponta como JavaServer Faces [8] ou Ajax [9];

- Lucene [10] - biblioteca *open source* de motor de pesquisa de texto de alta performance, escrita na linguagem de programação Java. Pode ser utilizada por qualquer plataforma que precise de indexar texto e de funcionalidades de pesquisa.

3.1.3 Trabalho realizado

Um dos principais objectivos deste projecto é de optimizar o processo de gestão de conteúdos online. Cada utilizador poderá ou não, consoante o seu nível de privilégios, editar conteúdos, através de um editor optimizado WYSIWYG. Os conteúdos editados entram num processo de aprovação que é configurado ao nível do conteúdo. Com um CMS, diferentes utilizadores deverão conseguir criar ou gerir conteúdos no portal, sem ser necessário conhecimento técnico (por exemplo gerar páginas Web sem saber a linguagem HTML).

A ideia inicial era construir um portal, tendo como base um CMS, onde diferentes utilizadores iriam ver, ou gerir conteúdo, consoante os seus privilégios. Este *conteúdo* pode ser interpretado como um parágrafo de texto, uma página Web, fotografias, *Portable Document Format* (PDF), apresentações, folhas Excel, recursos da Siemens, etc.

Dois domínios de interacção foram definidos (ver figura 3.1):

- Domínio Público - conteúdo geral e informação útil que pode ser acedida através da Siemens intranet;

- Domínio Privado (*back-office*) - onde várias tarefas administrativas são executadas, tais como: edição e validação de conteúdo que irá ser disponibilizado mais tarde no domínio público, gerir privilégios dos utilizadores, etc.

Para a construção deste portal *web* começou-se então por realizar um levantamento dos requisitos obrigatórios e opcionais. Após a definição dos requisitos, fez-se uma análise de gestores de conteúdos com o objectivo de escolher a melhor opção para integrar no portal em construção. A esta altura, já se tinham reunido as condições necessárias para começar a construir um protótipo. No entanto, para a realização deste protótipo, seria preciso seguir determinadas regras. Decidiu-se então, produzir umas *guidelines* para aplicações *web* para documentar essas regras e ajudar, tanto na realização deste protótipo, como também noutros projectos desenvolvidos por outras equipas da NSN. Depois de se ter desenvolvido o protótipo,

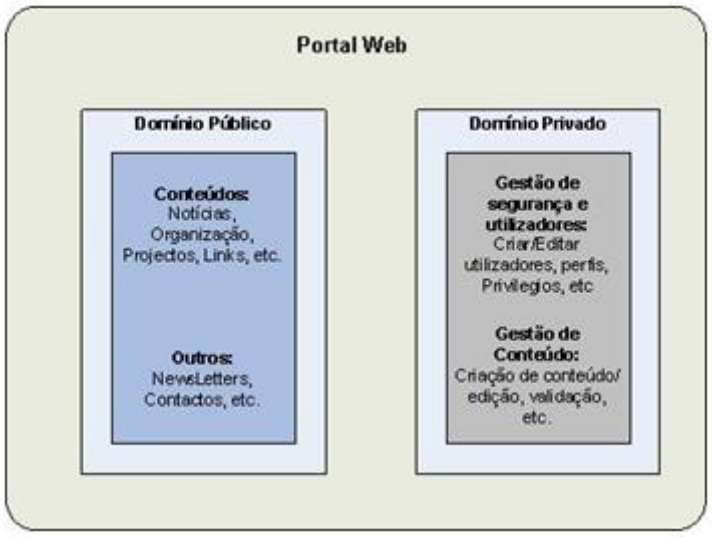


Figura 3.1: Domínios do portal Web

analisou-se editores WYSIWYG, motores de busca e aplicações para testes de regressão sendo, todos eles, requisitos obrigatórios para o projecto.

Todo este trabalho realizado é apresentado, detalhadamente, nas secções seguintes.

3.1.3.1 Levantamento de Requisitos

A primeira tarefa a realizar neste projecto foi o levantamento de requisitos. Para isso, foi executada uma análise a um total de 645 CMS e foram identificados os requisitos comuns. Fez-se o levantamento destes CMSs e foram-se aplicando filtros, tais como linguagem de programação, *open source*, etc., até se ficar apenas com um. O passo seguinte foi discutir quais os requisitos mandatórios e opcionais, através de reuniões com os responsáveis do projecto. De seguida, serão apresentados todos esses requisitos organizados por áreas:

- *System Requirements* (Requisitos do Sistema)

Requisitos	Descrição	Classificação
Base de Dados	O motor de base de dados usado para guardar o conteúdo deve ser eficiente e escalável (SQL Server ou Oracle)	Mandatório
Licença	Se um CMS já existente for usado, a sua licença tem de estar de acordo com as exigências da Siemens	Mandatório
Linguagem de programação	A linguagem de programação deve permitir um rápido, mas flexível, desenvolvimento (C# / ASP.NET ou Java / JSP)	Mandatório
Servidor Web	Esta decisão influencia o tipo de servidores em que o Web Portal possa correr (Windows / Linux). IIS ou Apache Web Server?	Mandatório

Tabela 3.1: *System Requirements*

- *Support* (Suporte)

Requisitos	Descrição	Classificação
Ajuda *Online*	Sistema de ajuda construído no CMS	Mandatório
Manual de Utilizador	Fornecer um manual de utilizador detalhado	Mandatório
Application Programming Interface (API)	*Templates* de código para ajudar utilizadores a desenvolverem novas funcionalidades	Mandatório
Manual de Desenvolvimento	Especificação do desenvolvimento	Mandatório
Testes de regressão	Automatizar testes de regressão	Mandatório
Treino	Fornecer ou comprar treino para os utilizadores do portal principal	Opcional
Comunidade de *Developers*	Se um CMS é usado, uma comunidade de *developers online* será uma grande ajuda	Opcional

Tabela 3.2: *Support*

- *Security* (Segurança)

Requisitos	Descrição	Classificação
Audit Trail	Detecta quem fez *updates*, quem adicionou ou apagou conteúdo	Mandatório
Aprovação de conteúdos	Permitir a avaliação de conteúdos	Mandatório
Privilégios	Privilégios de leitura e escrita por página ou conteúdo, bem como outros privilégios para outras funções do sistema	Mandatório
Autenticação LDAP	Autenticação LDAP para o acesso de utilizadores Siemens	Mandatório
Sandox	Área privada para os gestores de conteúdo submeterem novas ideias sem estarem preocupados com o funcionamento do resto do *site*	Mandatório
SSL Support	HTTPS para *logins*	Mandatório
Verificação de Email / Recuperação de Passwords	Chave de activação para os utilizadores, para assegurar que um endereço válido de *email* foi introduzido	Mandatório
Plugin de autenticação	Permitir a introdução de novos plugins com outros processos de auteticação	Opcional
Versioning	Gestão das versões dos conteúdos	Opcional

Tabela 3.3: *Security*

- *Ease of use* (Facilidade de Utilização)

Requisitos	Descrição	Classificação
Editor WYSIWYG	Editor de texto *web-based*, que permita criar conteúdo formatado sem a necessidade de se saber HTML, CSS, XML, XSL	Mandatório
Friendly URLs	Fornecer URLs legíveis	Mandatório
Mass Upload	*Upload* ou importar imagens e outros ficheiros de uma só vez	Mandatório
Undo	Operações de *undo* que permita apagar ou recuperar um conteúdo específico	Opcional
Conteúdo *Drag-N-Drop*	Permitir a operação de *drag-n-drop* de conteúdos	Opcional

Tabela 3.4: *Ease of Use*

- Performance

Requisitos	Descrição	Classificação
Replicação da Base de Dados	Replicação da base de dados pra uma melhor escalabilidade	Opcional

Tabela 3.5: Performance

- *Management* (Gestão do Sistema)

Requisitos	Descrição	Classificação
Administração *Online*	Gerir todo o sistema através de um *web browser*	Mandatório
Gestão de *Templates*	Estilos e *templates* para definir o *design* e o *layout*	Mandatório
Clipboard	Sistema que permita aos utilizadores a função de *cut and paste* de conteúdo	Mandatório
Motor de *Workflow*	Sistema de *Workflow* integrado no CMS para que possa ser usado na gestão de *business process* ou noutras tarefas além de aprovação de conteúdo	Mandatório
Asset Management	Repositório central para *uploads* de imagens e outros ficheiros, para permitir a reutilização dos mesmos em todo o portal	Opcional
Content Scheduling	Adicionar ou remover conteúdo, automaticamente, numa data pré-programada	Opcional
Themes / Skins	Mecanismo para transportar estilos, *templates*, etc, entre *sites*, para que se possa criar e reusar *themes*	Opcional
Lixo	Permitir aos administradores recuperar conteúdo que tenha sido removido	Opcional
Inline Administration	Permitir editar conteúdos directamente na página em que vai ser publicado	Opcional

Tabela 3.6: *Management*

- *Flexibility* (Flexibilidade)

Requisitos	Descrição	Classificação
Perfil de Utilizador Extensível	Fornecer uma *interface* administrativa para que se possa acrescentar novas propriedades ao perfil do utilizador	Mandatório
Reutilização de Conteúdo	Permitir reutilização de conteúdo em vários locais	Opcional
Localização da *Interface*	Permitir a tradução do portal para outras línguas, consoante a sua localização	Opcional
Metadata	Propriedades para todos os conteúdos, tipicamente usada para perfis, indexação, pesquisa. Exemplo: A *tag* Metadata em HTML	Opcional
Conteúdo multi-linguagem	Suportar a criação de *sites* com diversas linguagens	Opcional

Tabela 3.7: *Flexibility*

- *Built-in Applications* (Integração de Aplicações)

Requisitos	Descrição	Classificação
Forms Customizáveis	Mecanismo pra criar *forms* customizáveis	Mandatório
Gestão de documentos	Gestão *offline* de armazenamento e de versões de documentos	Mandatório
NewsLetter	Suportar *newsletters*	Mandatório
Votações	Suportar votações	Mandatório
Motor de Pesquisa	Integrar um motor de pesquisa que consiga indexar conteúdo e que permita o utilizador procurar esse conteúdo	Mandatório
Site Map	Gerar automaticamente a árvore do portal, mostrando todas as páginas	Mandatório
Fórum	Suportar um fórum para discussões	Opcional
Calendário de eventos	Apresentar eventos num caledário	Opcional
Gestão de eventos	Criar eventos e permitir utilizadores associarem-se a esses eventos	Opcional

Tabela 3.8: *Built-In Applications*

3.1.3.2 Análise de Gestores de Conteúdos

Depois de definidos os requisitos, passou-se ao próximo passo: Encontrar um CMS (*open-source* ou comercial) que disponibilizasse todos, ou a maior parte, dos requisitos. Para realizar esta tarefa com sucesso, foi feito um levantamento das características de vários CMSs (um total de 645). Após este levantamento, foram-se aplicando filtros (correspondem aos requisitos definidos anteriormente) um a um por ordem de importância, até se chegar a dois CMSs - Alfresco e DotNetNuke (apresentação realizada à equipa disponibilizada em anexo no Apêndice A).

O Alfresco [11] é um CMS *open source*, que é mais focado em organização de documentos do que em criação de aplicações Web. Desenvolvido na linguagem de programação JAVA, junta tecnologias de inovação *open source* com a estabilidade de uma plataforma registada, como por exemplo o Microsoft® SharePoint®. Por ser *open source*, permite ao Alfresco usar tecnologias também *open source*, usar contribuições de comunidades *open source* para atingir uma melhor qualidade de software e permite uma produção rápida do *software* e com baixos custos. Os objectivos principais são fornecer um CMS gratuito e, ao mesmo tempo, oferecer as mesmas características, funcionalidades e benefícios de outros CMSs pagos (Documentum ou Microsoft® SharePoint®). A interface gráfica do Alfresco é apresentada na figura 3.2.

Depois de devidamente instalado e analisado, o Alfresco apresentou as seguintes vantagens:

- Licença *open source*
- Application Programming Interface (API) bem definida e fornecimento de documentação JAVA para os programadores
- O motor de definição de regras para conteúdos específicos é potente e versátil, permitindo definir inúmeras regras.
- Bom sistema de controlo de versões de conteúdos
- Fácil de usar, com uma interface muito simples
- *Workflow* robusto
- Possibilidade de executar pesquisas avançadas sobre os conteúdos
- Boa comunidade de suporte
- Possibilidade de integrar o Alfresco com aplicações em *Hypertext Preprocessor* (PHP) e em .NET
- Permite às empresas, integrar este CMS com outros sistemas

E apresentou as seguintes desvantagens:

- Os ficheiros fonte não são fornecidos
- Não permite publicações de páginas Web
- Funcionalidades para integrar não são gratuitas
- *Drag and drop* em páginas Web não é permitido

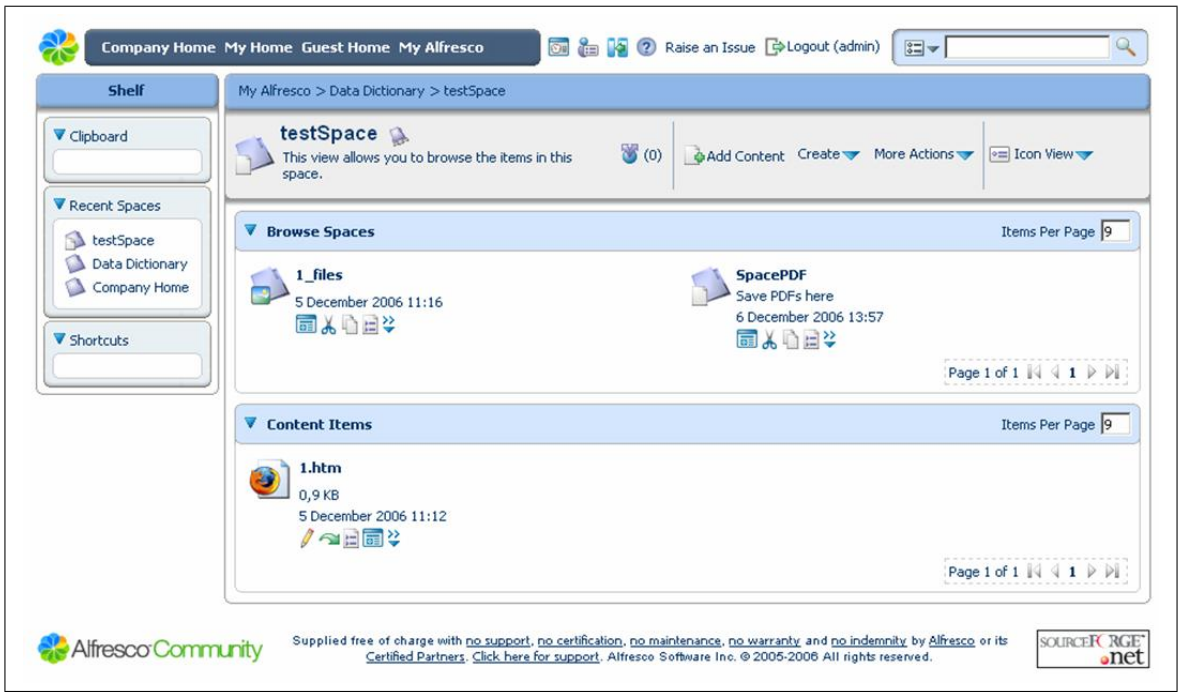


Figura 3.2: Alfresco *screenshot*

- *Sitemap* limitado
- *Skins* não disponíveis
- Uniform Resource Locators (URLs) gerados não são intuitivos

O DotNetNuke [12] é um CMS *open source* ideal para criar aplicações Web (páginas Web, portais de publicações *online*, etc.). Desenvolvido na linguagem de programação *Active Server Pages* (ASP.NET), a licença que o DotNetNuke oferece, garante que se possa obter este CMS sem custos. A interface gráfica do DotNetNuke é apresentada na figura 3.3.

Após uma cuidada análise do DotNetNuke, as vantagens inferidas foram as seguintes:

- Licença *opensource*, permitindo aos programadores mudar o código fonte consoante as suas necessidades
- Comunidade de suporte muito activa
- Facilidade na criação de páginas Web
- Suporta múltiplos *child portals*
- Pode ser estendido com módulos gratuitos

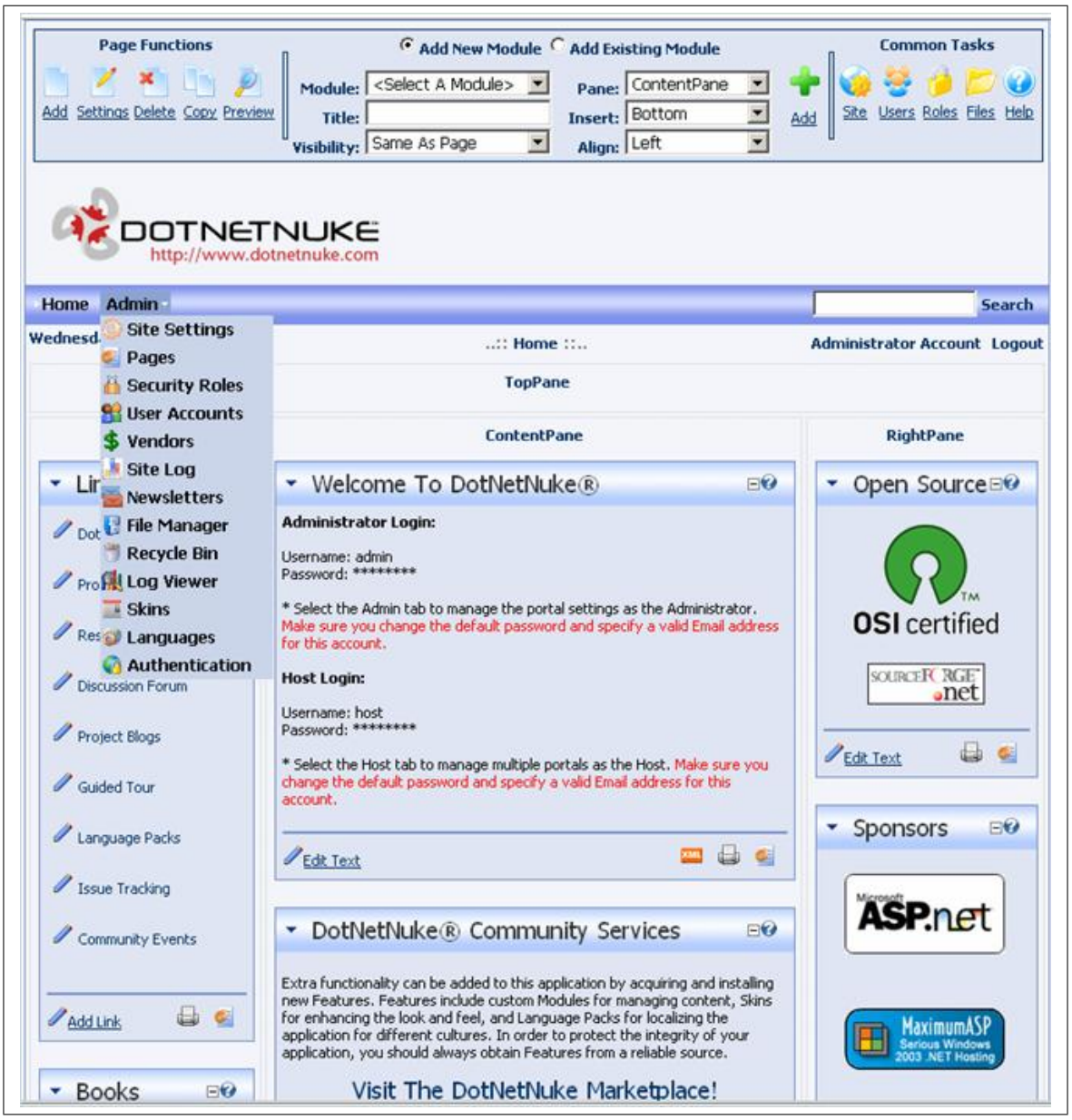


Figura 3.3: DotNetNuke *screenshot*

- Pode servir como base para páginas web de comércio electrónico, hospedar galerias de fotos, etc.
- Elevado número de *skins* disponíveis
- Possibilidade de fazer *drag and drop* de componentes na página Web

E as desvantagens:

- API não está devidamente documentada
- Documentos confusos que tentam explicar como adicionar módulos

- Muito tempo dispendido a tentar adicionar uma nova funcionalidade ao DotNetNuke através do código fonte
- Não tem biblioteca de documentos com controlo de versões
- Procura dentro de documentos não é possível
- Falha de segurança porque permite fazer o *download* de documentos directamente

Depois de apresentadas as vantagens e desvantagens de cada CMS (Alfresco e DotNetNuke), chegou-se a uma conclusão de qual dos dois se deveria propor para utilizar neste projecto. O CMS proposto foi o Alfresco porque cobria os requisitos mais importantes levantados anteriormente. Quanto ao DotNetNuke, teria sido uma boa opção se o projecto se focasse mais na criação de aplicações Web. A gestão de documentos similar à do Microsoft® SharePoint®, o motor de workflow e o controlo de versões foram factores primordiais para esta escolha.

3.1.3.3 *Guidelines* genéricas de GUI para web

Antes de se começar a desenvolver o protótipo pensou-se que seria importante perceber quais as *guidelines* necessárias, de modo que a GUI fosse construída de um modo consistente e coerente. As *guidelines* produzidas permitem:

- analisar a estrutura funcional da aplicação do ponto de vista do utilizador;
- obter informação sobre o *look and feel* das aplicações NSN;
- consultar regras especiais de *design* tais como, alinhamento de *text boxes*, cores, botões, etc.;
- consultar regras de usabilidade para assegurar que a aplicação a gerar seja fácil de usar.

O documento produzido é uma versão preliminar de GUI *guidelines* para aplicações *web*. Foi dividido em quatro capítulos: geral, navegação, *layout* e elementos de design.

As *guidelines* gerais estão relacionadas com aspectos gerais de aplicações *web*:

- suportar *eXtensible Hypertext Markup Language* 1.0 (XHTML) ou HTML 4.01, com o intuito de serem completamente compatíveis com uma das versões permitindo compatibilidade com os *browsers* mais usados. Ao usar uma destas versões, deve-se verificar o código com os *World Wide Web Consortium* (W3C) *validators* [13] - XHTML *validator* e HTML *validator*,

- páginas *web* devem suportar CSS, para garantir compatibilidade entre browsers e usar W3C *validators* para verificar o código - CSS *validator*,
- Incluir a declaração DOCTYPE em todas as páginas (ou na página *template*). Se se usar XHTML 1.0:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Usando HTML 4.01:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

- Indicar o *character set* em todas as páginas (ou na página *template*), usando o *Unicode UTF-8 character set*:

```
<meta http-equiv="Content-Type" content="text/html;charset=UTF-8">
```

- Usar caracteres especiais para garantir compatibilidade entre *browsers*(3.9)

Name	Symbol	Code
Ampersand	&	&amp;
Left Double Quote	“	&ldquo;
Right Double Quote	”	&rdquo;
Less Than	<	&lt;
Greater Than	>	&gt;
en-dash	-	&endash;
em-dash	–	&emdash;
Ellipsis	…	&helip;

Tabela 3.9: Caracteres Especiais

- Suportar *browsers* - garantir que as aplicações trabalhem nos *browsers* mais usados (*Internet Explorer* e *Mozilla Firefox*); não usar as últimas versões do

browser para desenvolvimento (usar *Internet Explorer* 6.0 e *Mozilla Firefox* 1.0); cuidado ao desenvolver as páginas *web* de modo que sejam suportadas pelos *browsers*, porque suportar outros *browsers* é sempre uma vantagem,

- Manter tempo de resposta do sistema em mente - manter tempo de *loading* no mínimo, evitando usar imagens de grande dimensão (a não ser que sejam mesmo importantes) e *Flash* (podendo ser usados se não existir outra maneira para fazer o que se pretende); dar *feedback* para operações que durem algum tempo através de *progress bars*; usar *scripts (JavaScript)* para aumentar a *performance* do servidor, testando sempre nos *browsers* suportados,

- Novas tecnologias como AJAX realçam a *user experience* - só deverão ser usadas quando trouxerem benefícios (difícil desenvolvimento); quando utilizadas, deverão ser testadas nos *browsers* suportados,

- Internacionalização (se aplicável) - colocar todas as *labels* botões, texto, etc, em ficheiros fonte separados; fornecer aos utilizadores a sua linguagem, permitindo trocá-la a qualquer momento; fornecer funções que permitam calcular a data da localização corrente do utilizador; alocar espaço para diferentes traduções de texto.

De seguida serão apresentadas *guidelines* que dizem respeito à navegação, por parte do utilizador, na aplicação *web*:

- Nunca retirar o *header* do *browser* que contém os botões de *Back* e *Forward*, sendo o botão *Back* mais frequentemente usado em todos os *browsers*,

- Permitir aos utilizadores retrocederem na sua navegação sem utilizar o botão do *browser*, permitindo assim, evitar *re-posts* ou perda de dados,

- Nos formulários, fornecer os botões *<action>* (informa qual a acção que o utilizador está prestes a realizar - *Login, Delete, Edit*, etc.), *Clear* (apaga todos os campos do formulário, se aplicável) e *Cancel* (cancela o formulário e volta para a página anterior),

- Nos formulários, associar o botão de omissão, a acções positivas e nunca a acções negativas. Por exemplo, uma confirmação de *Delete* nunca deve ter a opção *Yes* por omissão,

- Usar navegação de páginas em tabelas que mostrem muitos elementos (vinte ou mais) - Por exemplo: «Previous 1 2 3 4 5 6 7 8 9 10 Next »; *Previous* só será visível se a página anterior existir e *Next* só será visível se a página seguinte existir; Carregando em 10 a tabela desloca-se para a esquerda nove posições seleccionando o elemento 10,

- Fornecer uma navegação guiada e simples para tarefas mais complexas (semelhante a *wizards*) - identificar cada passo; apresentar aos utilizadores toda a sequência de passos; fazendo *highlight* do passo corrente; permitir aos utilizadores uma navegação entre os passos anteriores e posteriores; apresentar um sumário de todos as opções tomadas pelo utilizador antes de se efectuar a confirmação da transacção,

- Permitir aos utilizadores ordenar colunas de tabelas - através de um *click* no *header* da coluna; identificar a coluna ordenada e a direcção da ordenação com um pequeno ícone,

- Disponibilizar pistas de navegação - identificar cada página com um título no *header*; deixar que os utilizadores percebam em que página se encontram, fornecendo *breadcrumbs*; fornecer ajuda contextual em cada página; manter o botão de *help* visível numa posição consistente (sugestão: *top/right*),

- Evitar o uso de *pop-ups* - quando os utilizadores querem abrir um *link* numa nova janela, irão abrir uma nova página; fornecer mensagens de *feedback* (informação e erros) numa posição específica do ecrã; usar mensagens de confirmação para as perguntas de *Yes* ou *No* (Exemplo: *Do you want to delete XPTO?*),

- Manter o aspecto visual, usado frequentemente, dos *links* - os utilizadores devem distinguir, facilmente, *links* de elementos que não sejam *links*; usar o sublinhado e a cor; não sublinhar texto normal para não confundir os utilizadores; mudar a cor para os *links* visitados ajuda os utilizadores a identificar quais as páginas que já visitaram,

- Identificar o alvo de cada *link* - o texto do *link* deverá ser entendido sem o contexto, por exemplo, em vez de se usar *Click Here*, usar *Download PDF*,

- Fornecer um mecanismo de pesquisa eficiente e flexível (se aplicável) - motores de pesquisa devem suportar *queries* que incluam tipos, plurais, hífenes, etc.; resultados deve ser ordenados por relevância; a pesquisa deve ser apresentada numa caixa de texto simples localizada do lado esquerdo de um botão com a *label* igual a *Search* e não *Go* ou *Enter*,

- Nunca usar ficheiros do tipo PDF como conteúdo (por exemplo, páginas de ajuda) - PDFs quebram o fluxo de navegação por parte do utilizador, logo deve-se fornecer todo o conteúdo em HTML,

- Fornecer *tooltips* pequenas mas que descrevam o conteúdo de forma correcta - ícones, imagens, *links* e botões.

As *guidelines* produzidas, relacionadas com o *layout* de aplicações *web*, são as seguintes:

- *layout* da página - estas devem ser visíveis nos *browsers* mais modernos (*Internet Explorer, Mozilla Firefox, Opera, Netscape*, etc.); o *design* deve ser optimizado para a resolução 1024x768; adicionalmente, deve ser flexível o suficiente para adaptar a maiores resoluções (tirando partido do espaço vazio) e deve ainda suportar a redução da resolução para 800x600 (com a mínima usabilidade); evitar o *scroll* horizontal, porque perde-se muita informação que costuma estar visível para os utilizadores; restringir o *scroll* vertical para o máximo de duas vezes a altura da página,

- *Look and feel* consistente - usar o mesmo *look and feel* e *layout* em todos os ecrãs, o mesmo se aplica a elementos visuais, como os botões, texto, *links*, etc. Cores podem ser usadas para distinguir secções específicas, permitindo manter o significado e comportamento dos elementos,

- Usar CSS - permitem uma separação clara entre a funcionalidade e o design da aplicação *web*, fornecendo benefícios de manutenção da aplicação, e permitindo actualizar o *look and feel* com uma simples mudança,

- Usar *master pages* ou *templates* - uma única página define o *layout* fixo da aplicação *web*, permitindo às equipas de desenvolvimento, mudar o *layout* editando um único ficheiro,

- Não usar *frames* - não são bem suportadas por todos os *browsers*, em que o botão *Back* pode falhar em alguns desses *browsers*,

- Fornecer fontes legíveis - seleccionar uma fonte visível com um tamanho apropriado (*Verdana, Helvetica, Arial, Sans-Serif*); não usar um tamanho fixo (pixels) para o texto, de modo a permitir aos utilizadores com dificuldades visuais aumentar o tamanho da fonte da página; usar um contraste correcto, por exemplo, cores escuras num fundo claro; nunca usar cores claras para o texto, tal como o amarelo,

- Pontuação e letras maiúsculas - a primeira letra de qualquer *label* ou texto é sempre maiúscula. Todas as outras palavras são escritas em minúsculas, excepto nomes próprios e abreviações reconhecidas. Normalmente, os textos são alinhados à esquerda sem pontuação terminal, seja ponto final ou vírgula. Existe uma excepção se o texto consistir em mais de uma frase, sendo cada uma delas terminada por um ponto final,

- Não usar animações - estes elementos animados distraem os utilizadores das suas tarefas. Assim, deverá evitar-se tudo o que se assemelhe a anúncios,

- Permitir fácil impressão da página - garantir que todo o conteúdo das páginas possam ser imprimidos sem se perder conteúdo. Para tal, deverá usar-se CSS para remover todo o conteúdo desnecessário.

Por fim, as *guidelines* responsáveis pelos elementos de *design* são apresentadas de seguida:

- Botões de função - todos os botões de confirmação deverão estar alinhados à esquerda. Assim, permite à interface adaptar-se a resoluções maiores sem mudar a posição dos botões. Se o botão estiver associado a apenas um elemento, então deverá ser alinhado à direita no fim do elemento de design (por exemplo, *label* de um botão de uma *textfield*). Colocar os botões que são usados para confirmações (*OK, Cancel,* etc.), no mesmo nível horizontal, mantendo uma distância apropriada entre eles. Os botões para decisões positivas deverão ser colocados à esquerda dos botões para decisões negativas; fornecer uma *tool tip* para cada botão onde for tecnicamente possível e fornecer teclas de *shortcut* para botões frequentemente utilizados (usando JavaScript),

- *Labels* - fornecer *labels* para descrever o significado de áreas da página; colocar *labels* alinhadas à esquerda dos campos para *input*; nos *radio buttons* e *check boxes* estão posicionadas à direita; o último caracter das *labels* deverá ser o ':', com a excepção de a *label* corresponder a uma pergunta (usar o caracter '?'); marcar todos os campos obrigatórios com o caracter '*' antes da *label* (* *Second label*); usar letra maiúscula apenas na primeira palavra de qualquer *label*; todas as outras palavras são escritas em minúsculas, excepto nomes próprios e abreviações reconhecidas, como representado na figura 3.4,

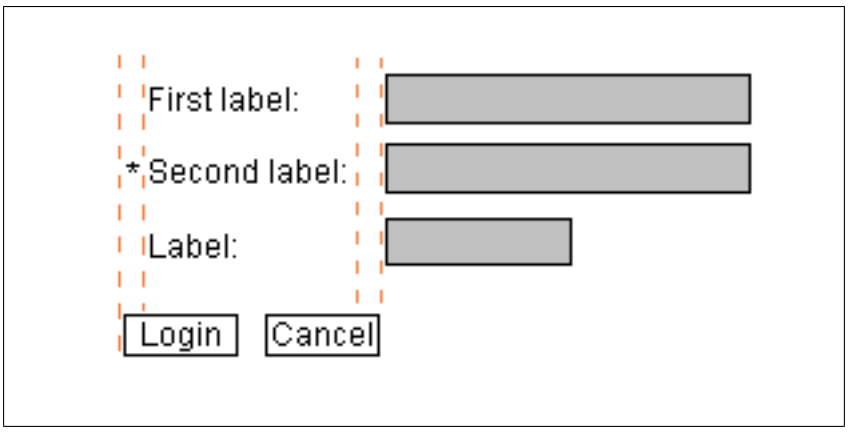


Figura 3.4: Alinhamento de *Labels* e campos

- Tamanho dos campos de inserção de texto consoante o conteúdo esperado,
- Fornecer ajuda para campos de inserção de texto que necessitem de um formato específico - por exemplo, quando é preciso inserir uma data, deve-se indicar qual o formato (exemplo: dd/mm/aa) à direita do campo de inserção de texto, como ilustrado na figura 3.5,

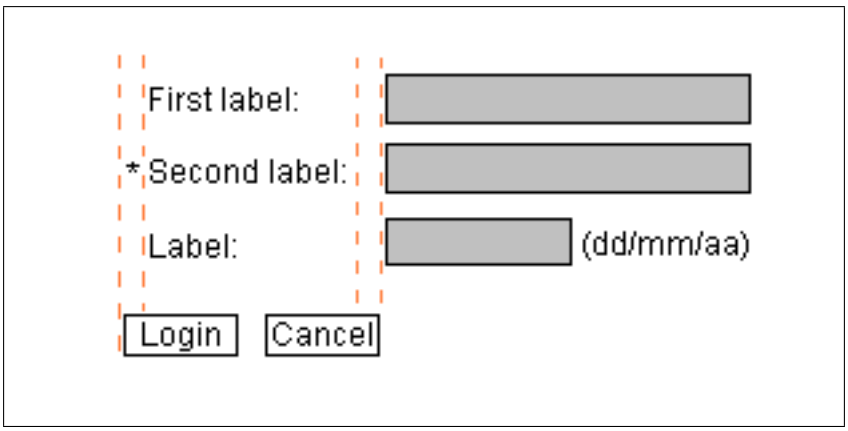


Figura 3.5: Informação de ajuda para campos de inserção de texto

- Fornecer mensagens de ajuda - podem ser de três tipos: mensagens de erro, de informação/aviso e de confirmação; mostrar as mensagens de um modo consistente; as mensagens de erro e de aviso devem ser apresentadas numa posição específica do ecrã (sugestão: no topo e à esquerda dos conteúdos); nos formulários, quando devolvem uma mensagem de erro, os campos devem ser assinalados com um símbolo diferente (exemplo: mão vermelha a indicar o erro), enquanto as mensagens de confirmação deverão ser apresentadas numa página de confirmação.

Depois de produzidas as *guidelines* passou-se ao próximo passo - o desenvolvimento do protótipo. Durante a construção do protótipo, estas *guidelines* foram de extrema utilidade, sendo consultadas sempre que se deparasse com uma dúvida de *design*.

3.1.3.4 Protótipo de Baixa-Funcionalidade

Após a participação em várias reuniões com os responsáveis do projecto, a interface gráfica foi definida e, a partir daí, foi construído um protótipo (HTML/JavaScript), restrito às guidelines da Siemens, apenas para o GUI de *front-office*. Este permitiu aos responsáveis do projecto visualizar e interagir com um protótipo funcional do site, no ecrã. Outro objectivo deste protótipo foi tentar perceber se a navegação entre páginas era adaptada aos utilizadores e se o uso de *breadcrumbs* (técnica

de navegação que exibe todas as páginas visitadas até à página corrente) oferecia benefícios aos utilizadores finais. Depois de tratados os últimos pormenores, através de discussões com os responsáveis, chegou-se ao protótipo final, apresentado na figura 3.6.

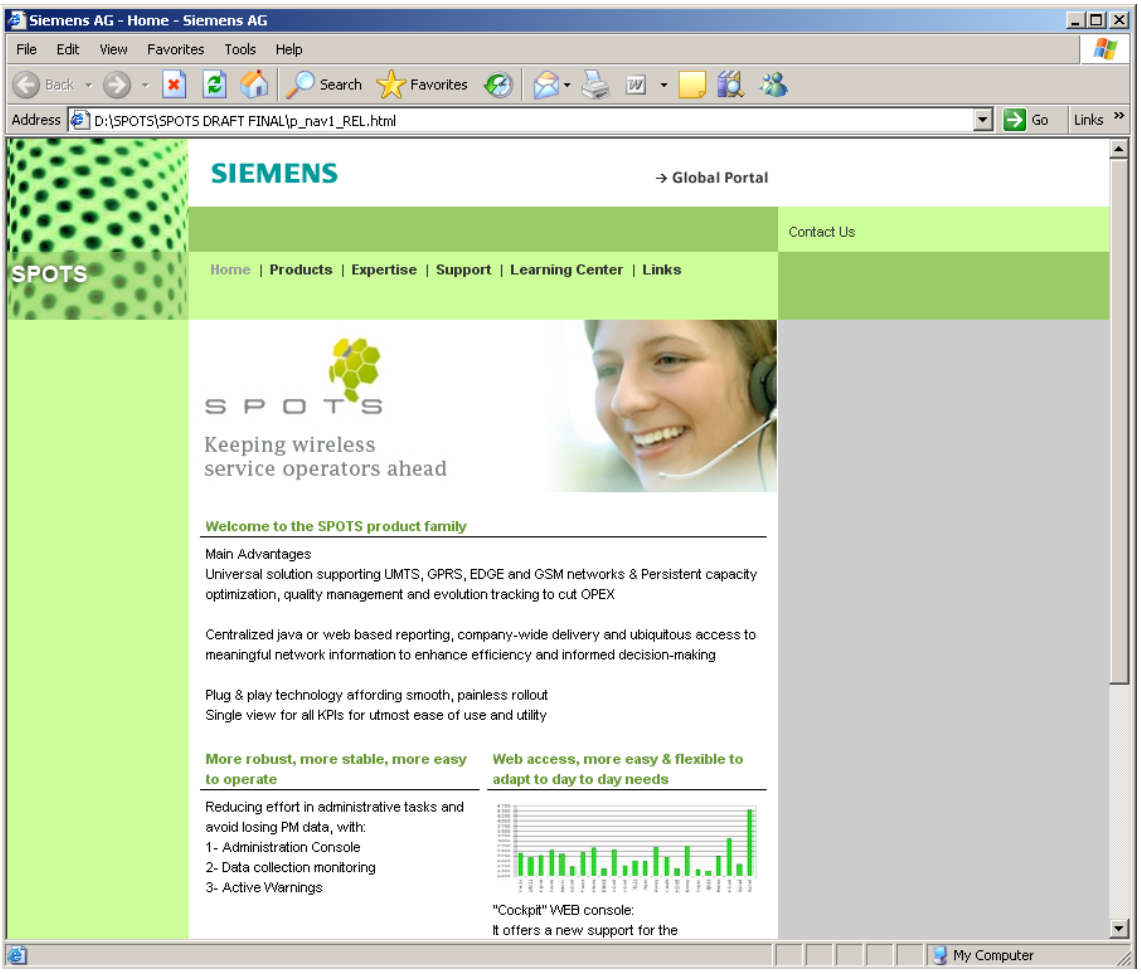


Figura 3.6: Protótipo

3.1.3.5 Análise de Editores WYSIWYG

A utilização de um editor WYSIWYG é um requisito mandatório para este projecto. Actualmente, a maior parte dos editores têm várias limitações, entre elas:

- o editor não permite que o conteúdo seja editado eficientemente;
- o conteúdo editado não corresponde às expectativas do utilizador;
- alguns utilizadores, como não conseguem editar o conteúdo como querem, passam a editar directamente em linguagem HTML.

Como é óbvio, os utilizadores do portal Web não têm que aprender uma nova ferramenta para editarem o conteúdo. É fácil perceber que os editores actuais não cumprem este objectivo. Os utilizadores esperam destes editores aquilo que têm no Microsoft Word, que é uma aplicação a que estão perfeitamente ambientados. Devido às limitações, foi realizada uma investigação para encontrar alternativas ao uso de editores WYSIWYG. Com base neste conceito, foi desenvolvido um protótipo que está actualmente em análise e que poderá resultar numa patente NSN.

3.1.3.6 Análise de Motores de Busca

Um dos requisitos mandatórios do projecto era integrar um motor de pesquisa que conseguisse indexar conteúdo e que permitisse que esse conteúdo fosse encontrado pelo utilizador, através de uma pesquisa. Para este requisito ser cumprido, uma análise aos motores de busca existentes foi realizada com o objectivo da integração do mesmo no projecto. O primeiro passo foi fazer uma pesquisa e levantamento da maior parte dos motores de busca existentes. De entre todos estes motores de busca só os *open source* é que foram considerados - um total de 37. Filtros como a linguagem de programação, o sistema operativo sobre o qual o motor de busca funciona e a possibilidade de personalizá-lo (desenvolvendo novas funcionalidades a partir do código fonte), entre outros, foram aplicados. Apenas três motores de busca "sobreviveram"a estes filtros - DocSearcher [14], Regain [15] e Zilverline [16] (apresentação realizada à equipa em anexo no Apêndice D).

O DocSearcher é uma aplicação *desktop* de pesquisa que usa, como base, a biblioteca de pesquisa Lucene, funcionando em qualquer plataforma que tenha a tecnologia JAVA instalada. As pesquisas são realizadas sobre um *index* (conceito usado para representar a pasta onde são colocados todos os ficheiros, sobre os quais se pretende efectuar uma pesquisa). O DocSearcher permite criar vários *indexes* e permite efectuar pesquisas em ficheiros HTML, *Text format* (TXT), PDF, *Rich Text Format* (RTF), Word, Excel, Open Office. Os ficheiros do tipo *Zone Information Protocol* (ZIP), *Roshal Archive* (RAR) e *PowerPoint* (PPT) não são suportados.

Esta aplicação permite indexá-los, mas não realiza a pesquisa nesses ficheiros, não devolvendo, assim, qualquer tipo de resultado. Quando a palavra a pesquisar existe em algum dos ficheiros existentes no *index*, o DocSearcher devolve os resultados por ordem em termos de relevância (número de vezes que a palavra a pesquisar existe num determinado ficheiro). Na figura 3.7 é apresentada a interface gráfica desta aplicação:

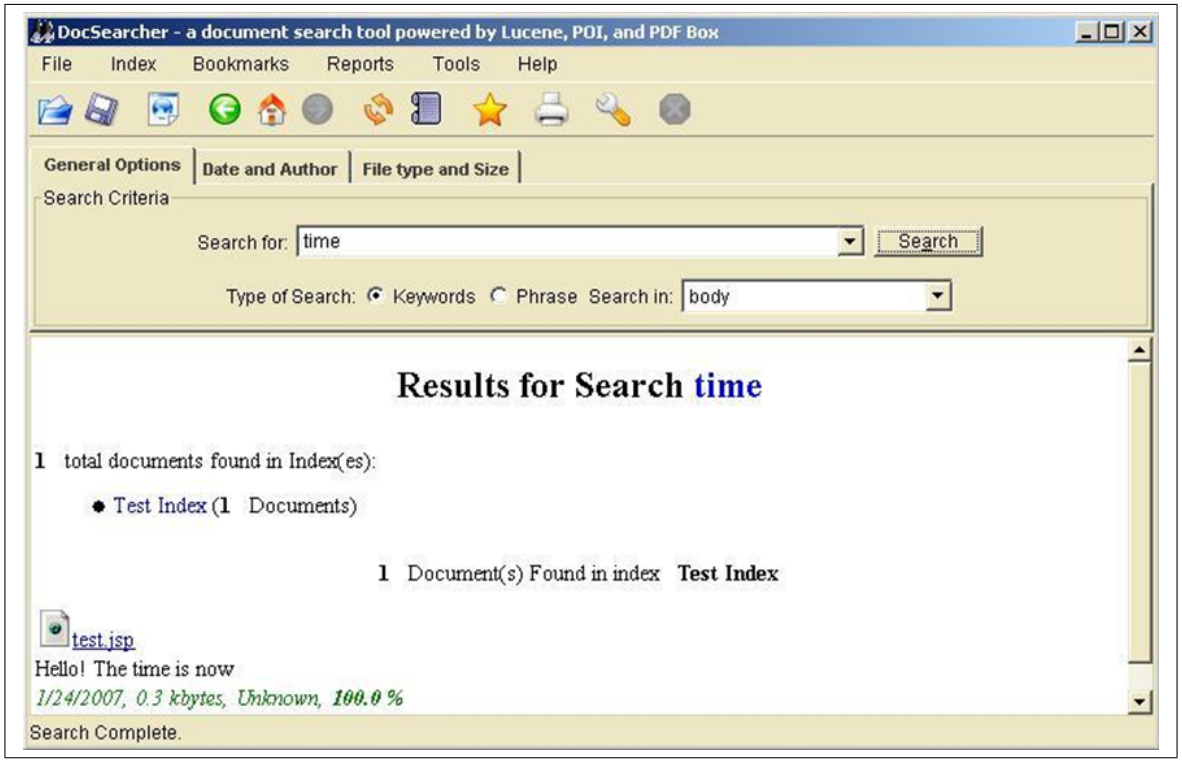


Figura 3.7: DocSearcher

O Regain é um motor de pesquisa semelhante a outros motores de pesquisa Web, como o Google, mas em vez de se pesquisar na Web, pesquisa-se em *indexes* criados previamente pelo utilizador. O Regain é baseado, também, na biblioteca Lucene para criar e procurar palavras nos ficheiros inseridos nos *indexes*. É escrito em JAVA, logo qualquer plataforma que suporte esta tecnologia, suporta também este motor de busca. O Regain consegue pesquisar em ficheiros do tipo HTML, TXT, PDF, RTF, textitWord Document (DOC), Excel (XLS), PPT, *JavaServer Pages* (JSP). Na figura 3.8 é apresentada a interface gráfica deste motor de busca:

O Zilverline é o terceiro motor de busca analisado, apresentado na figura 3.9. Oferece um acesso tipo Web, ao conteúdo pessoal. Tal como o DocSearcher e o Regain, o Zilverline é escrito em JAVA e é baseado também na biblioteca de pesquisa Lucene. Este motor de pesquisa permite extrair conteúdo de ficheiros do tipo PDF, DOC, XLS, PPT, RTF, TXT, JAVA, *Microsoft Compiled HTML Help* (CHM), ZIP e RAR, conseguindo indexar e procurar os mesmos, em inúmeras directorias.

Figura 3.8: Regain

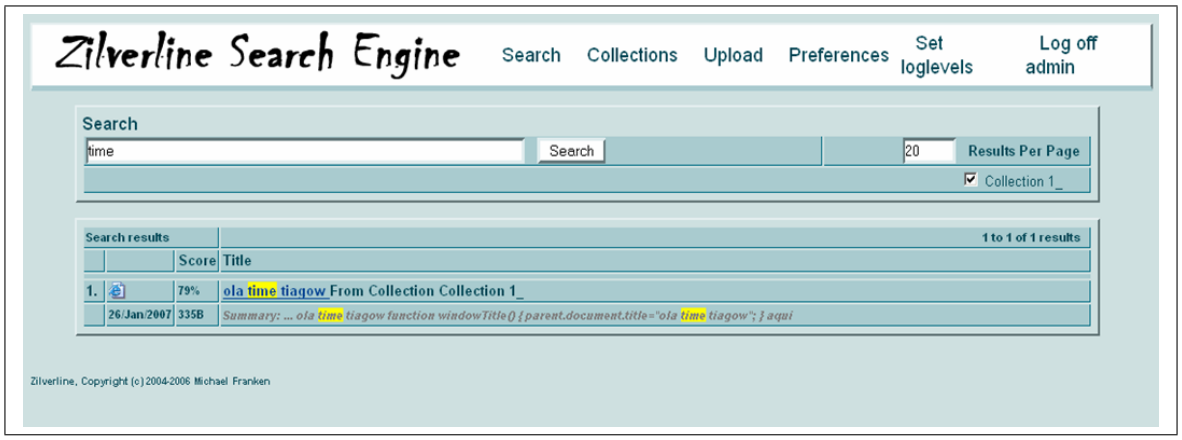


Figura 3.9: Zilverline

A escolha entre estes três motores de busca não foi fácil. As diferenças entre as funcionalidades de cada um são ínfimas, mas um teria de ser escolhido para fazer parte do projecto. Para ajudar nesta escolha foi realizada uma tabela (3.10) com as características mais importantes:

Após uma análise cuidada da tabela 3.10, foi proposta a utilização do DocSearcher como motor de busca do projecto. Dentro das características apresentadas na tabela, os factores que mais pesaram na exclusão do Zilverline e do Regain, foi a licença (não é gratuito para uso comercial) e a complexidade do código (elevada complexidade a comparar com o DocSearcher e Zilverline), respectivamente.

	DocSearcher	Regain	Zilverline
Indexing Time	Fast	Fastest	Slow
Indexing Rules	Flexible	More Flexible	Flexible
Indexing Creation (Difficulty)	Easy	Easiest	Easy
Filetypes Supported	HTML, PDF, RTF, TXT Word, Excel	HTML, PDF, RTF, TXT Word, Excel, Powerpoint	HTML, PDF, RTF, TXT Word, Excel, Powerpoint, ZIP, RAR
Highlighting	No	No	Yes
Several Indexes Creation	Yes	Yes	Yes
Source code complexity	Normal	High	High
Advanced Search	Yes	Yes	No
License	GPL	LGPL	Collaborative Source License

Tabela 3.10: Comparação entre Motores de Busca

3.1.3.7 Padrões de Desenho

Antes do começo da execução das tarefas referentes ao projecto, concluiu-se que uma aprendizagem pormenorizada sobre *Design Patterns* (Padrões de Desenho") seria muito útil. Durante o tempo de execução do projecto, muitos dos problemas que se iriam enfrentar, iriam ser problemas comuns. Os responsáveis pelo projecto disponibilizaram todos os meios necessários para que essa aprendizagem fosse bem assimilada e realizada com sucesso. Após um estudo intensivo da documentação, e depois de assimilar os novos conhecimentos sobre Design Patterns, uma apresentação em PPT foi realizada e exibida aos restantes colegas da equipa (encontra-se em anexo no Apêndice B).

Esta aprendizagem teve como principal objectivo relembrar quais os padrões de desenho existentes e em que casos específicos se deveriam utilizar - uma grande ajuda para o desenrolar do projecto.

3.1.3.8 Análise de Aplicações para Testes de Regressão

Uma aplicação para automatizar testes de regressão é outro requisito mandatório para o projecto. Esta análise foi feita em conjunto com outro colega, que analisou uma aplicação que apresentou várias limitações: o *qftestJUI* [17]. Em alternativa, analisou-se a aplicação WinRunner [18] que é uma ferramenta de testes funcionais e de regressão, que reduz o tempo de teste e optimiza o esforço do teste de uma aplicação. Foi desenvolvida uma aplicação muito simples em Java Swing, para testar as funcionalidades do WinRunner, a partir de um plano de testes realizado para uma aplicação da Siemens. Para criar um teste basta gravar uma sequência de acções

sobre a aplicação, por exemplo, o utilizador carregar num botão, escreve numa caixa de texto e escolhe um *item* numa *combo box*, etc. Durante a gravação, existe a opção de poder editar uma acção directamente no script gerado. Pode-se ainda adicionar checkpoints para comparar o resultado esperado e o resultado corrente. Há a possibilidade de, por exemplo: comparar imagens, verificar transacções em bases de dados, ordenar tabelas Excel, etc. Por estas razões, chegou-se à conclusão que esta aplicação poderá ser bastante útil para optimizar o tempo de teste de aplicações.

3.1.4 Resultados

Os resultados deste projecto não foram os que se esperavam, devido à sua interrupção, não tendo a equipa nem o estagiário qualquer responsabilidades nesta decisão. O esperado, seria realizar com sucesso cada tarefa correspondente ao projecto até ao seu final. As tarefas que ficaram por realizar foram, a integração dos CMS e motor de pesquisa escolhidos, Alfresco e DocSearcher respectivamente, no protótipo de baixa funcionalidade desenvolvido e a implementação de requisitos não suportados após a integração. No entanto, muito trabalho foi realizado até o projecto terminar.

Para o projecto Ferramentas de Suporte à Gestão de Conteúdos, através de várias reuniões com os responsáveis do mesmo, foram levantados todos os requisitos obrigatórios e opcionais (apresentados amteriormente) permitindo concluir quais os mais prioritários: um editor WYSIWYG que permita ao utilizador editar conteúdos sem ter de saber uma linguagem técnica para tal; uma API para poder extender a ferramenta a especificidades da equipa; possibilidade de editar privilégios por utilizador, por página, por conteúdo ou funcionalidade; gestão de de estilos e *templates* para definição de *layout*; *workflow* que permita a uma pessoa, com maiores responsabilidades, aprovar ou rejeitar conteúdos propostos; motor de pesquisa.

O Alfresco foi o CMS escolhido após um levantamento de todos os CMSs existentes, para se integrar no portal *web* a desenvolver. Após esse levantamento e após a aplicação de inúmeros filtros, apenas três sobressaíram.

Antes de se passar à construção do protótipo, no qual seria integrado o CMS Alfresco, foi produzido um documento de *Guidelines* de GUI para *web*. Este foi criado com o intuito de ajudar a interiorizar essas *guidelines* para as pôr em prática no próximo passo - a construção do protótipo. Este protótipo foi construído em HTML e Java Script tendo como base as *guidelines* produzidas anteriormente. À medida que se ia construindo este protótipo, teve de se participar em reuniões com os responsáveis do projecto para ultimar uns pormenores, com vista a se chegar o mais rapidamente possível à versão final do mesmo (demonstrado na secção Protótipo de Baixa-Funcionalidade).

Outro requisito obrigatório seria incluir um motor de busca neste projecto. Tal como nos CMSs, fez-se também um levantamento de motores de busca *open source*,

sobrando apenas três no final da aplicação de determinados filtros. Após uma análise cuidada dos três (secção Análise de Motores de Busca), foi escolhido o DocSearcher para se integrar no projecto.

O próximo passo seria integrá-los uns nos outros, mas tal já não foi possível. Com a incerteza sobre o futuro do projecto, o desenvolvimento parou e não houve oportunidade de acabá-lo como estava previamente planeado, apesar de todos os objectivos intermédios terem sido atingidos com sucesso. No futuro, se o projecto recomeçar, a equipa UXG poderá continuar a desenvolvê-lo.

3.2 Suporte à Gestão de Projectos

O projecto "Suporte à Gestão de Projectos"será apresentado, detalhadamente, nesta secção, bem como o contexto onde se insere.

3.2.1 Contextualização

A NSN, devido à sua dimensão, está envolvida em inúmeros projectos nas mais diversas áreas de negócio. Tipicamente, um projecto está associado a uma equipa com vários elementos, em que cada um está associado a uma ou mais tarefas desse mesmo projecto. Antes de se dar início, o esforço de cada pessoa numa ou mais tarefas é calculado para verificar quantas são necessárias para que o plano do projecto não se atrase e se torne exequível. A boa gestão das pessoas de uma equipa, associadas às tarefas de um projecto, é um aspecto fulcral para que o mesmo seja executado dentro dos prazos pré-estabelecidos. Assim, o contexto deste projecto passa, através de uma ferramenta de Sistemas de Informação(SI) de BPR já existente, o *Eagle v1.0*, por monitorizar e apoiar tarefas de projectos de *Network Management*. Esta nova funcionalidade permitirá verificar, graficamente, não só se determinadas tarefas de um dado projecto estão ou não atrasadas, relativamente ao plano, como também a extensão de cada tarefa até ao fim do projecto, dado o esforço actual. Se a tarefa de um dado projecto estiver atrasada e se o esforço dedicado a essa tarefa se mantiver, esta nova funcionalidade permitirá constatar a discrepância entre o esforço planeado e o esforço actual. Com estes dados, apresentados graficamente, o responsável pela gestão do projecto consegue obter uma visão global do esforço dedicado a cada tarefa, verificando a extensão das mesmas até ao fim do projecto.

3.2.2 Tecnologias e Ferramentas

Para tornar possível a concretização deste projecto, foram utilizadas inúmeras tecnologias. Algumas destas tecnologias já eram familiares, enquanto que, para outras, foi necessário um tempo de aprendizagem. De seguida, é apresentada uma breve

descrição das tecnologias utilizadas no projecto:

- IBM Rational Portfolio Manager (RPM) [19] - é uma solução em tempo real de gestão de projectos e portfolios, fornecendo visibilidade sobre todas as Tecnologias de Informação (TI) e permitindo o alinhamento de *software* e desenvolvimento de sistemas com os objectivos de negócio. Esta tecnologia usa os dados de projectos para análises, relatórios e construção de gráficos, automatizando a recolha desses dados, de modo a minimizar erros. Serve principalmente para atribuir recursos a projectos e a planear futuras necessidades, de uma forma intuitiva e eficiente;

- IBM Rational ClearCase [20] - é uma ferramenta de controlo de versões e de revisão de código de todo o ciclo de desenvolvimento. Gere ficheiros, directorias e outros recursos de desenvolvimento, tendo o controlo sobre *workspaces* pessoais, permitindo o acesso exacto às versões necessárias desses recursos. O desenvolvimento paralelo e a gestão de *builds* e *releases* são outras funcionalidades suportadas por esta ferramenta. Desta forma, o Rational ClearCase, através do desenvolvimento paralelo, das *builds* e *releases* em tempo reduzido e da reutilização de código, aumenta a produtividade;

- Java - Linguagem de programação já descrita no projecto anterior;

- JSP [21]- É uma tecnologia Java que permite um rápido desenvolvimento de aplicações Web. Gera, dinamicamente, HTML, XML ou outros tipos de ficheiros, quando um pedido de um determinado cliente *Web* é executado. Esta tecnologia separa a UI da geração dos conteúdos, permitindo assim aos *developers* poderem alterar a interface sem terem de interagir com o conteúdo dinâmico, ou alterarem o conteúdo dinâmico sem terem de mudar o *layout* da página *Web*;

- *Extensible Markup Language* (XML) [22] - linguagem simples e muito flexível que contém informação estruturada hierarquicamente, facilitando a partilha de dados entre diferentes plataformas ou sistemas, via *Internet*;

- *Simple Object Access Protocol* (SOAP) [23] - protocolo responsável pela troca de mensagens XML, via *HyperText Transfer Protocol* (HTTP), sobre determinada rede de computadores. O padrão de mensagens mais utilizado por este protocolo é o *Remote Procedure Call* (RPC), que consiste no envio de uma mensagem por parte de um *network element* (cliente), para outro (servidor), respondendo este último, imediatamente, ao cliente;

- *Enterprise Java Beans* (EJB) [24] - É um componente *server-side* que permite que os *developers* criem elementos de *software* reutilizáveis. Esta tecnologia está dividida em várias camadas: camada de apresentação (*Servlets*, JSP, XML), lógica de negócio(os próprios EJBs do tipo sessão), camada de integração (EJBs do tipo entidade e *Java Database Connectivity* (JDBC)) e a própria base de dados. Esta divisão em camadas tem o intuito de fornecer um rápido desenvolvimento de aplicações Java em componentes distribuídos, transaccionais, portáveis e seguros. A arquitectura e os tipos de EJBs serão explicados com maior detalhe, mais à frente, no contexto da secção *Enterprise Java Beans* (apresentação disponível em anexo no Apêndice C);

- *Apache Ant* [25] - É uma ferramenta, baseada em java, que permite fazer o *build* de uma aplicação a partir de um ficheiro XML. Descreve o processo de *build* e as suas dependências e, por defeito, tem o nome de *build.xml*. Neste ficheiro podem-se definir inúmeros tipos de *tasks*, tais como, criar ou apagar directorias, compilar aplicações usando o Java compiler(*javac*), criar ficheiros do tipo Jar, War, etc. De seguida, na figura 3.10, um excerto de um ficheiro *build.xml* é apresentado:

```
<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
     - Compilation targets                                         -
     - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
<target name="clean" description="cleans the created directories">
  <delete includeemptydirs="true" quiet="true">
    <fileset dir="${target.dir}" />
    <fileset dir="${dist.dir}" />
  </delete>
</target>

<target name="compile">
  <mkdir dir="${target.classes.dir}" />

  <javac srcdir="${src.dir}" destdir="${target.classes.dir}"
         debug="${build.debug}" includes="${src.java.includes}"
         excludes="${src.java.excludes}">
    <classpath refid="classpath" />
  </javac>
</target>
```

Figura 3.10: Excerto de um ficheiro *build.xml*

- Apache Tomcat [26] - É um *web container open source* desenvolvido pela *Apache Software Foundation* que inclui um servidor HTTP interno, suportando *servlets* e JSPs, podendo funcionar também como um servidor *web* independente. Fornece um ambiente estável e seguro para que código Java, em conjunto com um *web server*, possa ser executado. Inclui ferramentas de con-

figuração e gestão de aplicações, sendo possível fazer o *deploy* das mesmas a partir de, por exemplo, um ficheiro XML;

- Apache Axis [27] - Ferramenta *open source* baseada em Java e XML desenvolvida pela *Apache Software Foundation*, que permite a construção de *Web Services*. Consiste na implementação de um servidor SOAP na linguagem de programação Java, e em APIs com o intuito de criar *Web Services*;

- *Structured Query Language* (SQL) [28] - Linguagem usada para criar, apagar, actualizar e obter dados de base de dados. Por exemplo, considerando uma base de dados com todos os colaboradores da NSN, se se pretender ir buscar toda a informação sobre o colaborador Tiago Honorato, o seguinte comando é executado:

  SELECT * FROM colaboradores WHERE nomeCompleto="Tiago Honorato";

  Este comando SQL obtém toda a informação da tabela *colaboradores* onde o campo *nomeCompleto* seja igual a Tiago Honorato;

- IBM Database (DB) 2 V8 [29] - Ferramenta, desenvolvida pela IBM, de gestão de bases de dados. Pode ser gerida através da linha de comandos, através de comandos próprios da ferramenta, necessitando um maior conhecimento desta ferramenta por parte do administrador. Por exemplo, para se ligar a uma base de dados *TESTE* com nome de utilizador igual a *admin* e password igual a *pass*, o seguinte comando teria de ser executado:

  db2 connect to TESTE user admin using pass;

  Como alternativa, existe um *Control Center* que disponibiliza uma GUI, permitindo gerir a base de dados, oferecendo inúmeras funcionalidades, tais como: ligar base de dados; criar base de dados; remover base de dados; linha de comandos própria para executar comandos SQL sobre determinada base de dados; criar *backups* de bases de dados; restaurar base de dados a partir de *backups*. O *Control Center* desta ferramenta é apresentado na figura 3.11.

- Jenia [30] - Biblioteca composta por inúmeras famílias de componentes JSF, que resolvem diferentes cenários *web* consoante a necessidade dos *developers*: *Dynamic*: Efeitos Dynamic Hyper Text Markup Language (DHTML) para as páginas JSF; *Chart*: Geração dinâmica de gráficos; *Popup*: Tratar vários tipos de *popups*; *DataTools*: extensão para JSF *DataTables*; *Template*: Mecanismo *template*;

- *Java API for XML-based RPC* (JAX-RPC) [31] - API usada para construir aplicações *web* e *web services*, incorporando funcionalidades RPC baseadas na linguagem XML de acordo com a especificação do protocolo SOAP;

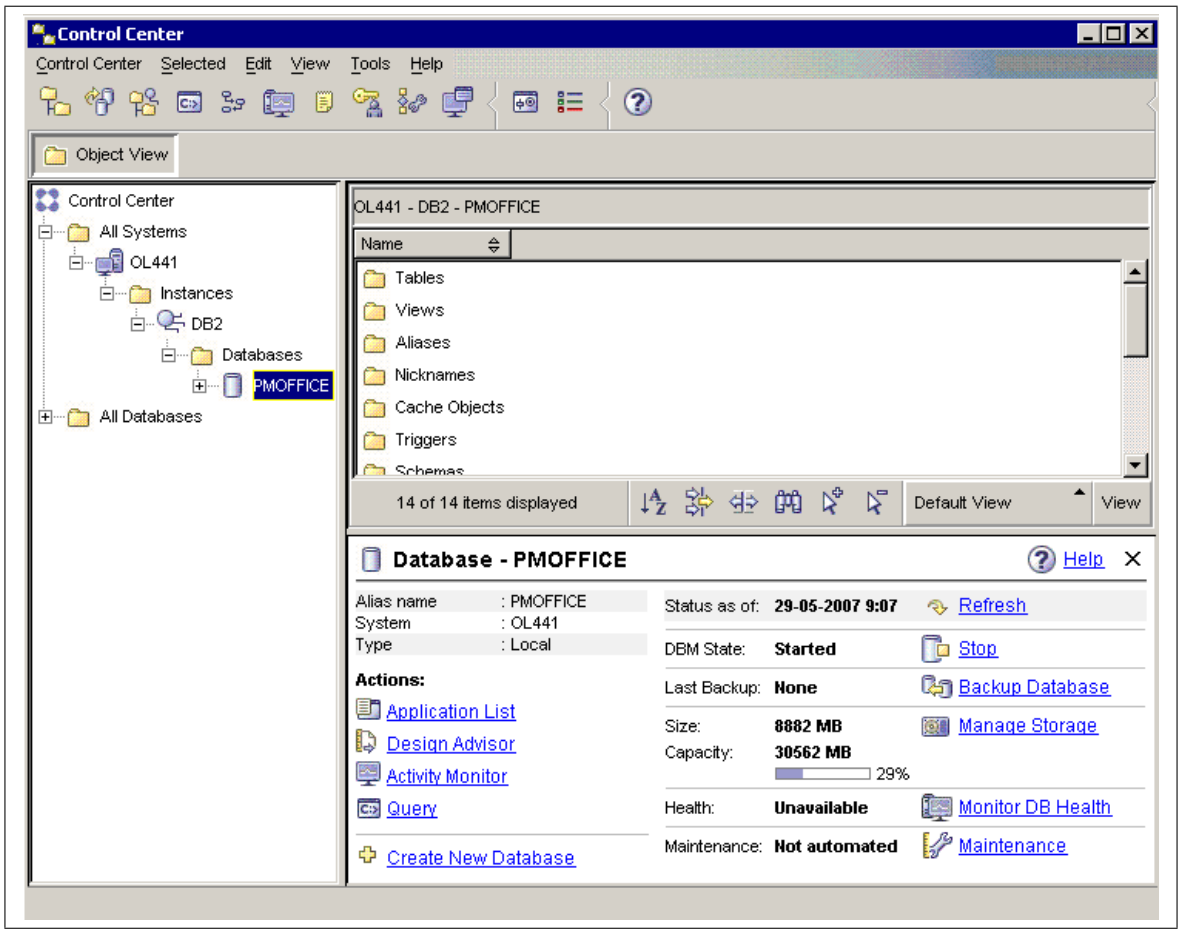


Figura 3.11: DB2 *Control Center*

- *Java API for XML-based Web Services*(JAX-WS) [32] - Tecnologia semelhante à JAX-RPC, mas mais recente, em que a principal diferença é o uso de *annotations* (Exemplo: @WebMethod). Ao usar *annotations*, o desenvolvimento é simplificado e o tamanho dos ficheiros *Java Archive* (JAR) reduzido;

- *Eclipse* [33] - é uma plataforma *open source* de desenvolvimento, escrita na linguagem de programação Java, usada para desenvolver, gerir, ou fazer *deploy* de software durante o seu ciclo de vida. Esta plataforma permite também incluir *plug-ins*, de uma forma muito simples, para fornecer mais funcionalidades;

- *Web Services* - Serviços acedidos através da *web*, usados por sistemas distribuídos para comunicarem entre si. Tecnologia descrita, detalhadamente, mais à frente;

- *Web Services Description Language* (WSDL) [34] - é uma linguagem baseada em XML que descreve *Web Services*, através de uma *interface* publica. WSDL é usado numa combinação de SOAP e XML para fornecer *Web Services* via

Internet. Um programa cliente que se ligue ao *Web Service*, consegue ler o WSDL e verificar todos os métodos que o servidor disponibiliza. Usando o protocolo SOAP, o cliente pode executar a chamada a um método que esteja listado no ficheiro WSDL.

3.2.3 Trabalho realizado

O principal objectivo deste projecto é obter uma visualização gráfica da monitorização de tarefas de projectos de *Network Management*, através de *Web Services*. Após uma primeira fase, apenas de pesquisa, sobre *Web Services*, passou-se a uma fase de testes. Estes testes, tiveram como objectivo não só obter uma familiarização sobre este conceito, como também, criar um *Web Service* muito simples, utilizando duas tecnologias diferentes (JAX-RPC e JAX-WS). Depois de ganhar experiência com *Web Services*, teve de se pensar se existiria alguma maneira de usar este conceito para aceder ao servidor de base de dados. Chegou-se à conclusão que seria possível, mas com um senão - teria de se gastar um esforço adicional a actualizar a ferramenta de gestão de base de dados (DB2) da versão 7 para a versão 8, e fazer uma migração da base de dados do RPM da versão 6.1.2.7 para a versão 7.0.1.1, pois só esta última versão é que suporta *Web Services*. Após a actualização da ferramenta de gestão de base de dados e a migração da mesma terem sido executadas com sucesso, foi realizado o próximo passo - aceder à base de dados via *Web Services* para recolher os dados necessários, com o objectivo de representá-los graficamente na ferramenta de suporte à gestão de projectos - o *Eagle v1.0*.

Para realizar este projecto com sucesso o estudo da arquitectura, de EJB, de *Web Services*, da DB2 e do próprio *Eagle v1.0* foram fulcrais.

3.2.3.1 Arquitectura

A arquitectura do projecto é apresentada na figura 3.12. Esta arquitectura divide-se em três grandes componentes - IBM RPM, *Eagle v1.0* e *Web Services*. A ferramenta RPM é baseada numa arquitectura de três camadas:

- Base de dados - a ferramenta RPM usa uma base de dados como um repositório para toda a informação que é gerida pelas aplicações. Esta camada suporta outra ferramenta, a IBM DB2, que é responsável pela gestão da base de dados;
- Aplicação - Esta camada trata de quase todo o processamento de dados. É a interface entre o cliente e a base de dados do RPM. A camada Aplicação é implementada usando os protocolos HTTP e *HyperText Transfer Protocol Secure* (HTTPS) com servlets. Os pedidos HTTP vindos do cliente são processados pelo *application server* que, por sua vez, submete um pedido à base de dados da ferramenta RPM. O pedido é executado, e as respostas enviadas

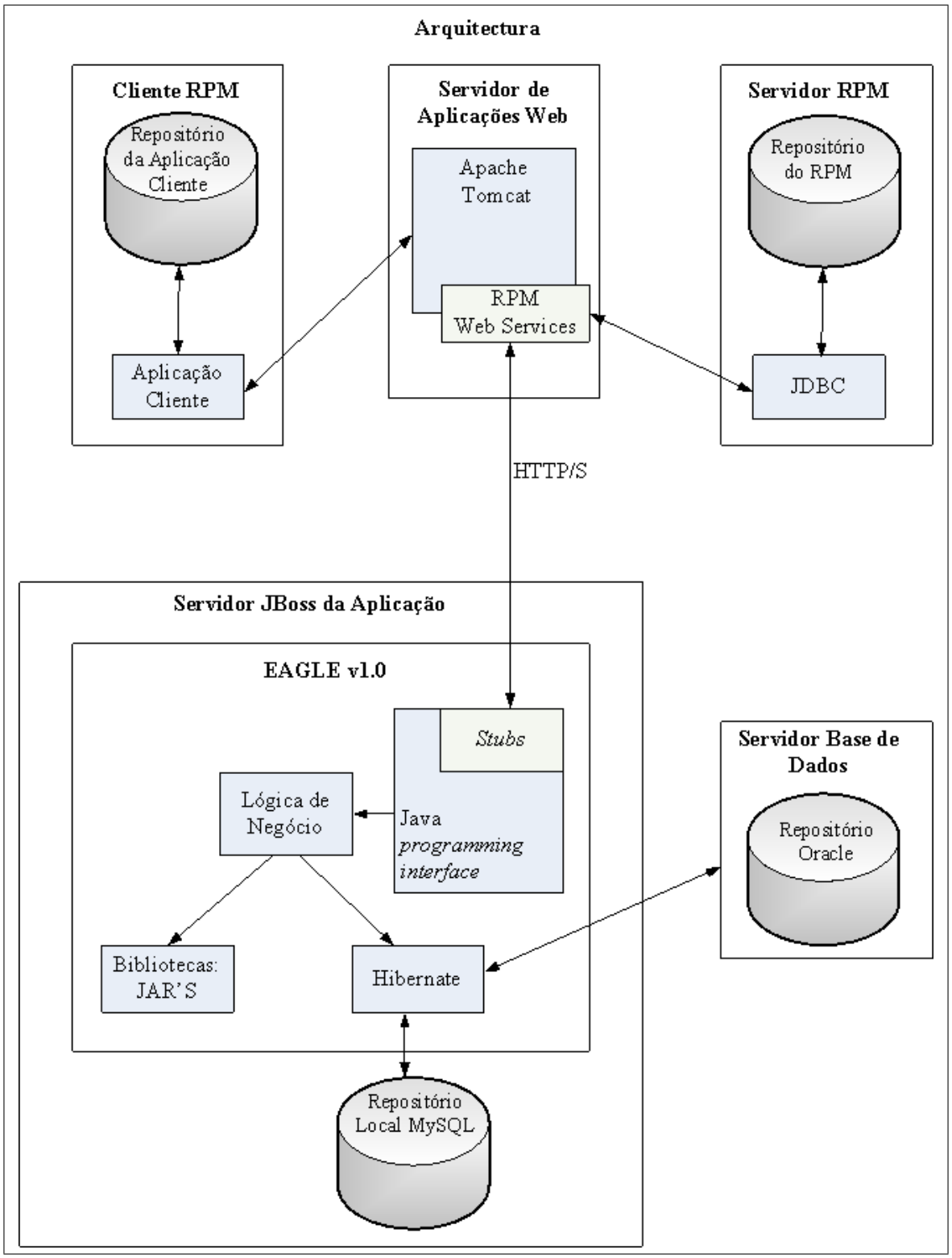


Figura 3.12: Arquitectura

pela base de dados são processadas pelo *application server*, sendo este último o responsável por enviar essas respostas para o cliente.

O RPM precisa de um *servlet container* (servidor *web* que suporte a execução de *servlets*), tais como *IBM WebSphere Application Server, Apache Tomcat* ou *WebLogic Server*. Duas *servlets* são importantes neste processo - *servlet* de apresentação e *servlet* de integração. A *servlet* de apresentação é responsável por tratar todos os pedidos HTTP para transacções ou *queries* no RPM. Esta *servlet* recebe e envia pedidos de dados via HTTP para os *stored procedures* [1] *e queries* alojados na base de dados, para que esses pedidos sejam executados com sucesso. Este procedimento só é possível porque esta *servlet* utiliza JDBC para se conectar à base de dados.

A *servlet* de integração é uma *servlet* Java que trata de todos os pedidos HTTP e XML para a integração de outros sistemas. Esta *servlet* funciona como intermediária entre esses sistemas externos e os serviços de integração. Tal como a *servlet* de apresentação, a *servlet* de integração também faz uso da JDBC para se ligar à base de dados.

- Cliente - o cliente RPM é composto por duas camadas. A primeira camada é a camada de apresentação, que representa a GUI com a qual o utilizador interage para manipular os objectos RPM. A segunda camada tem o nome de *broker*. Actua como se fosse uma camada intermédia entre a camada de apresentação (GUI) e o *application server*. Por isso, qualquer manipulação de dados executada na camada de apresentação será empacotada em pedidos pelo *broker* e submetidos pra o *application server*, sendo o inverso também verdadeiro. As respostas enviadas pelo *application server* são desempacotadas pelo *broker* e enviadas para a camada de apresentação.

A ferramenta de suporte à gestão de projectos - o *Eagle v1.0*, que corre sobre um servidor *JBoss*, é acessível, dentro da *Intranet* da NSN, através de qualquer um dos seguintes browsers: *Internet Explorer*, *Mozilla Firefox* ou *Opera*. O acesso à base de dados local *MySQL* é executado, em tempo real, através do *Hibernate* [35]. Esta base de dados local é preenchida, diariamente às 0 horas, com informação proveniente do servidor de base de dados *Oracle* [36] remoto. A actualização dos dados é realizada diariamente às horas referidas, com o objectivo de manter a base de dados local o mais actualizada possível e, ao mesmo tempo, evitar uma sobrecarga da mesma, permitindo assim, manter uma elevada velocidade de acesso.

A lógica de negócio da aplicação é responsável pela comunicação com a base de dados local, processando funcionalidades: oferecidas por bibliotecas (JARs) e previamente implementadas por outras equipas da NSN.

[1] Código compilado que reside num servidor de base de dados que reduz o processamento no cliente

Trata da transmissão dos pedidos do utilizador para a camada de acesso ao repositório local (*MySQL*) e também, do processamento da informação recebida da base de dados para a interface. A tecnologia EJB é usada nesta camada porque permite aos *developers* criarem pedaços reutilizáveis de *software*. Outra vantagem dos EJBs, que é usada pela lógica de negócio, é que permitem um simples e rápido desenvolvimento de aplicações distribuídas, seguras, persistentes, transaccionais, concorrentes, escaláveis e de confiança.

Os *Web Services* do RPM usam sessões para regular a troca de informação entre cliente (Sistema Cliente) e servidor (Servidor RPM). Uma sessão permite a um cliente autenticar-se, com o intuito de poder usar os serviços disponibilizados pelos *Web Services*, enquanto o servidor o permita. Assim, uma sessão não é mais do que um identificador, que é associado a um cliente quando este faz o *login*. Se este *login* for efectuado com sucesso o cliente pode efectuar as seguintes operações no repositório do Servidor RPM: criar, renovar e apagar dados; obter dados; pesquisas.

Para cada uma das operações, a aplicação cliente faz um pedido aos *Web Services* do RPM. Instalados num *Application Server*, que neste caso é o *Apache Tomcat*, acedem ao repositório do Servidor RPM via JDBC e enviam a resposta para a aplicação cliente processar os resultados. Os *Web Services* fazem *commit* automaticamente quando há uma alteração nos dados.

Esta arquitectura mostra a integração de chamadas a *Web Services* na ferramenta de suporte à gestão *Eagle v1.0*. Assim, é possível aceder de uma forma indirecta à base de dados. O uso de *Web Services* por parte desta ferramenta traz vantagens e desvantagens:

- Vantagens
  - Não exige uma compreensão exaustiva da arquitectura da base de dados, bastando conhecer os serviços disponibilizados;
  - Quando é lançada uma nova versão do produto (RPM) não será preciso ajustar sempre a nova base de dados com as *queries* pretendidas.
- Desvantagens
  - Com acesso directo à base de dados existe maior liberdade, podendo-se retirar da base de dados exactamente a informação que se pretende, não se limitando aos serviços disponibilizados;
  - Em termos de velocidade, o acesso directo é muito mais rápido.

Ao optar-se pela utilização de *Web Services* neste projecto pensou-se, não na velocidade de acesso à base de dados mas sim, na facilidade de implementação

aquando uma mudança de versão do RPM. Assim, os *Web Services* foram a tecnologia escolhida, disponibilizando todos os serviços necessários para realizar o projecto e prevenindo um elevado gasto de recursos se se optasse por outra tecnologia.

3.2.3.2 *Enterprise Java Beans*

Como já foi referido na secção Tecnologias e Ferramentas, um EJB é um componente *server-side* que permite aos *developers* criar elementos de *software* reutilizáveis. Na figura 3.13, é apresentada a arquitectura dos EJBs.

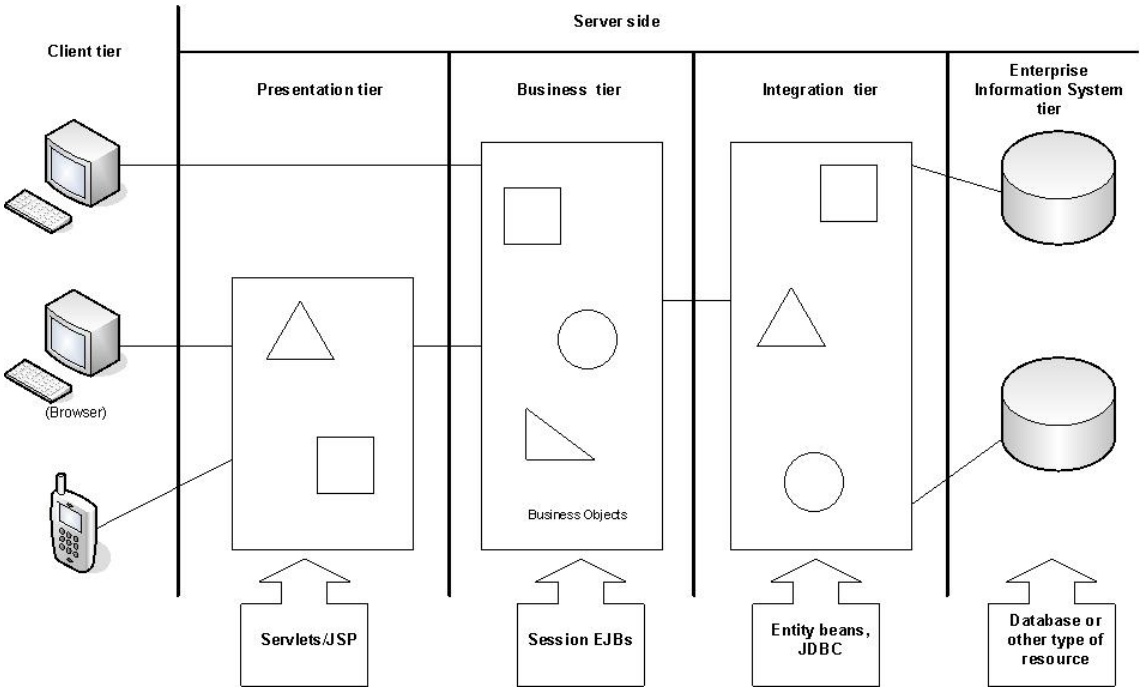


Figura 3.13: *J2EE Architecture*

A camada cliente é responsável pela lógica de apresentação através de vários tipos de cliente (clientes remotos, *browsers*, etc.), que poderão interagir com o servidor. A arquitectura está dividida em quatro camadas do lado do servidor:

- A camada de apresentação (*presentation tier*) inclui *servlets*, JSP, XML ou Extensible Style Language (XSL), permitindo existir uma separação entre a lógica de negócio e a lógica de apresentação, com o intuito de, se existir modificações da camada do lado do servidor, o cliente não precisará de fazer alterações;
- A camada de negócio (*business tier*) inclui os próprios EJB, onde a lógica de negócio é implementada;
- A camada de integração (*integration tier*) é a responsável pelo acesso e a actualização de vários recursos, geralmente implementados por *entity EJBs* (explicado mais à frente nesta secção);

- A camada *Enterprise Information System tier* (EIS) representa os dados através de uma base de dados.

Um *developer* que esteja a implementar uma aplicação distribuída sem esta tecnologia, terá de implementar a parte de segurança, transacções, persistência da informação, tratamento de excepções, sincronização, etc; existindo uma grande probabilidade de escrever código repetido. Assim, o *developer* gasta mais tempo a construir estas funcionalidades do que a desenvolver a lógica de negócio do EJB. A alternativa viável é atribuir as actividades referidas em cima para o *Java server-side component*. Este último é constituído pela combinação de EJB, EJB *container* e *application server* como demonstrado na figura (3.14).

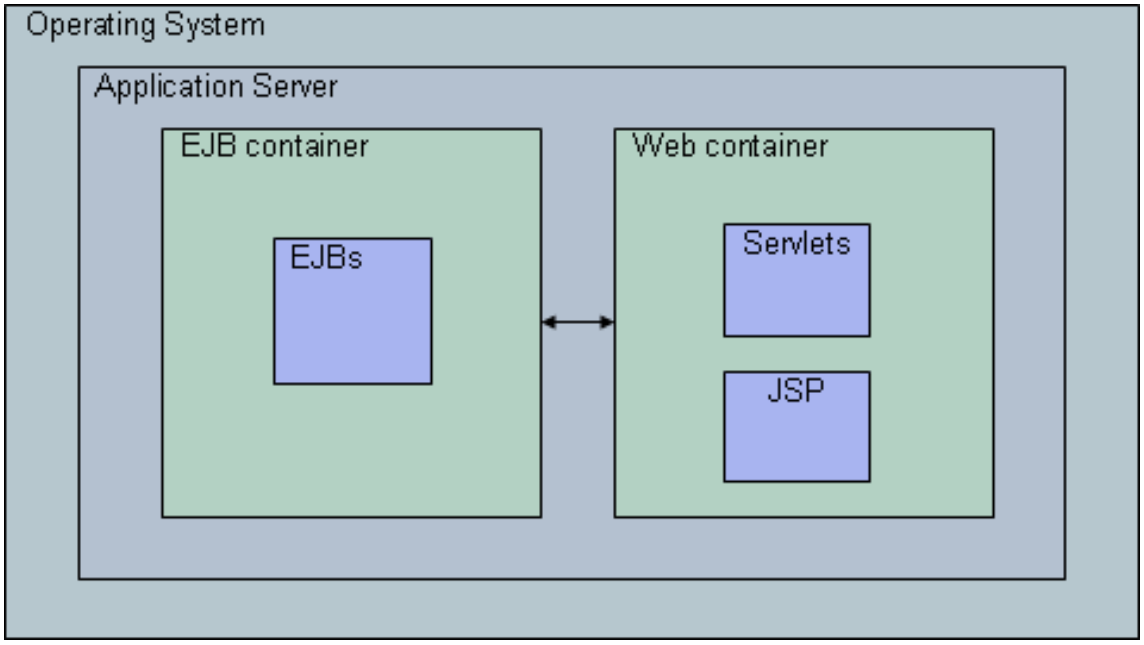


Figura 3.14: *Java server-side component*

Na estrutura os EJBs são responsáveis por fornecer a lógica de negócio e estão contidos num EJB *container* que tem a função de fornecer funcionalidades para a segurança, gestão de estados, *multi-threading*, transacções, etc. Este EJB *container* comunica com um *web container* que tem a função de permitir o acesso via *web* à camada de apresentação do EJB. A função do *application server* nesta estrutura é fornecer aos EJB e *web containers* os serviços indispensáveis, tais como: conectividade à rede, transacções distribuídas, *clustering*, etc.

Há três tipos de EJBs, os *Entity EJBs*, os *Session EJBs* e os *Message-driven EJBs*. Os *Entity EJBs* fornecem uma vista aos dados guardados no repositório, um acesso partilhado a múltiplos utilizadores e, desde que os dados se mantenham na base de dados, este tipo de EJBs podem ser de longa duração. Um *Entity EJBs* pode ser classificado como *Container managed persistence*(CMP) ou *Bean*

managed persistence(BMP). Nos CMPs o código que permite chamar funções ligadas às bases de dados (JDBC) são geradas automaticamente por este tipo de *bean*. As vantagens deste tipo de *Entity EJB* é q se pode mudar apenas a configuração da base de dados, o que permite que se escreva menos código, dando origem a um rápido desenvolvimento. Nos BMPs o *developer* é responsável por escrever a *interface* para a base de dados. Nunca é tão eficiente como os CMPs porque tem de se escrever muito código e por isso, a sua gestão poderá complicar-se, daí a sua utilização ser quase nula.

Os *Session EJBs* são os responsáveis pela implementação da lógica de negócio. Não representam dados partilhados directamente na base de dados, mas podem aceder e actualizar esses mesmos dados. Existem dois tipos de *Session EJBs*, os *Stateless Session EJBs* e os *Stateful Session EJBs*. Os *Stateless Session EJBs* não têm memória, ou seja, nunca se lembram do passo anterior. Fáceis de desenvolver e muito eficientes permitem servir múltiplos clientes. Estes EJBs poderão ser usados quando é necessário, por exemplo, um cliente fazer um pagamento de um conjunto de compras. Quando um cliente efectua o pagamento, o *bean* não precisa de qualquer dado desse mesmo cliente. Os *Stateful Session EJBs* têm memória, logo precisam de mais recursos e, ao necessitarem de mais recursos, tornam-se mais "pesados". Ao contrário dos *Stateless Session EJBs* estes EJBs só consegue servir um cliente de cada vez. Por exemplo, se se pretender representar um conjunto de compras, um *Stateless Session EJBs* seria o mais indicado. Este teria de se lembrar de todos os artigos desse conjunto de compras (associado a cada cliente) durante a sessão.

Por último, os *Message-driven EJBs*. São responsáveis por fornecer uma execução assíncrona da lógica de negócio, ou seja, quando um cliente executa uma chamada a um método, não terá de ficar à espera da resposta e poderá continuar a executar. Podem actualizar dados de uma determinada base de dados como por exemplo, poderão actualizar uma tabela de preços.

Neste projecto, o *bean* escolhido para representar os gráficos foi um *Session EJBs*, configurado como ilustrado na figura 3.15.

Tanto as funcionalidades já implementadas (gráficos de *history*, *times*, etc.) como esta nova funcionalidade que mostra o *effort* actual e planeado de um projecto, são geridas por outro bean - *statisticsProjectsBean*. Contém um método para cada tipo de gráfico existente, que devolve um *boolean* (verdadeiro se o utilizador, através da GUI, escolher um projecto e um gráfico, e falso se não escolher uma das opções anteriores), usado pelo ficheiro JSP que é responsável por gerar os gráficos pretendidos (usando a tecnologia Jenia).

A implementação deste *bean* foi concretizada no ficheiro *effortChartsBean*, onde são chamados os RPM *Web Services* e recolhidos os dados necessários para a construção dos gráficos, o qual será demonstrado (apenas excertos do ficheiro), uma

```
<managed-bean>
  <managed-bean-name>effortChartsBean</managed-bean-name>
  <managed-bean-class>
    com.siemens.bpr.userPortal.beans.effortChartsBean
  </managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
  <managed-property>
      <property-name>projectsBean</property-name>
          <property-class>
            com.siemens.bpr.userPortal.beans.StatisticsProjectsBean
          </property-class>
          <value>#{statisticsProjectsBean}</value>
    </managed-property>
</managed-bean>
```

Figura 3.15: Configuração do *bean*

parte na secção *Web Services* (o que diz respeito aos *Web Services*) e outra parte na secção Recolha de dados e Apresentação (o que diz respeito à apresentação dos gráficos resultantes).

3.2.3.3 *Eagle v1.0*

O *Eagle v1.0* é uma ferramenta de Sistemas de Informação (SI) de BPR que executa rastreios e monitoriza processos dentro de projectos de *Network Management*, ou seja, uma ferramenta muito útil para a gestão de projectos e de recursos (pessoas) associados a esses mesmos projectos.

Um módulo para incluir nesta ferramenta foi realizado, com o intuito de utilizar *Web Services* disponibilizados pelo RPM. Para se integrar este módulo no *Eagle v1.0*, a arquitectura do mesmo teve de ser estudada, tal como algumas tecnologias já referidas anteriormente.

Para realizar esta integração foi acrescentado um novo *bean* que representasse os gráficos a construir. Um *Session EJBs* foi integrado nesta ferramenta, como já referido na secção EJB. Anteriormente à integração deste módulo, todas as funcionalidades da ferramenta teriam de aceder directamente à base de dados. Com esta nova integração, a ferramenta de gestão de projectos tornou-se numa ferramenta mais flexível.

3.2.3.4 *Web Services*

Web Services permitem integrar aplicações baseadas num conjunto de protocolos e *standards*, tais como:

- XML - linguagem que define a estrutura e o conteúdo dos Web Services;

- (WSDL) - linguagem baseada em XML que é usada pra descrever as capacidades dos *Web Services*, permitindo aplicações comunicarem eficientemente. Um ficheiro WSDL não é mais que um ficheiro XML que define objectos e métodos;

- SOAP - protocolo baseado em XML usado para enviar mensagens via *Web Services*. Estas mensagens são enviadas entre a implementação cliente dos *Web Services*, e o SOAP *handler* no servidor do RPM.

Descrevem uma maneira de integrar aplicações Web sobre a Web, através de interfaces programáveis. Estas interfaces programáticas disponíveis são, na verdade, o próprio *Web Service*.

Web Services partilham a lógica de negócio, dados e processos através da sua API. Os *developers* de uma dada aplicação, podem adicionar o *Web Service* a uma interface (tais como uma página Web ou um programa executável) para oferecer uma funcionalidade específica aos utilizadores. É possível então que novas aplicações possam interagir com aquelas que já existem, e que sistemas desenvolvidos em linguagens diferentes ou em plataformas diferentes sejam compatíveis. A *World Wide Web*, é usada cada vez mais para comunicações entre aplicações.

A ferramenta RPM introduz os Web Services como a tecnologia preferida para permitir a integração entre aplicações externas e ela própria. Esta tecnologia permite aos clientes ligarem automaticamente as suas bases de dados com a base de dados do RPM e executar um carregamento inicial do repositório do RPM. Um acesso à base de dados, a partir de um programa Java, C♯ ou C++, usando a API dos *Web Services*, é possível. Com esta API é possível desenvolver um acesso directo à informação da base de dados do RPM, com o objectivo de criar inúmeras soluções de integração, usando as linguagens de programação já referidas. A grande vantagem dos *Web Services* do RPM consiste em construir aplicações cliente independentes da UI. Isto é, uma aplicação cliente não precisa de ser alterada sempre que a UI do RPM muda e evita a necessidade de apresentar os dados da mesma maneira que o RPM, beneficiando assim, as preferências do utilizador.

Dentro da integração da aplicação cliente, a interface dos *Web Services* do RPM é equivalente a um objecto da linguagem de programação da aplicação. Um cliente SOAP é usado para gerar interfaces de objectos de negócio (*business-objects*) e *stubs* de rede, a partir de um ficheiro WSDL que especifica o *schema* da mensagem RPM, o endereço do serviço e outras informações de interface. A integração da aplicação cliente trabalha com dados na forma de propriedades de objectos, e envia e recebe os dados por chamadas a métodos de um certo objecto. O cliente SOAP trata dos detalhes da construção do pedido SOAP e envia-o para o RPM, encaminhando a resposta para um objecto com o qual seja fácil trabalhar. Isto evita que os *developers*

precisem de fazer *build* e *parse* dos documentos XML, permitindo focarem-se apenas na gestão, e na apresentação dos dados.

Por simplificarem o acesso das aplicações aos dados do RPM, por serem seguros, e por permitirem uma programação da aplicação independente da interface do RPM, os *Web Services* são a melhor escolha para a comunicação entre aplicações externas e o próprio RPM.

Para permitir o uso dos RPM *Web Services* neste projecto, estes, através de um ficheiro de configuração XML, foram *deployed* num *application server* - o *Tomcat*. Nesse ficheiro foi configurado o tipo de conexão à base de dados, a sua localização e os *username* e *password* necessários para que a ligação à base de dados, anteriormente migrada, fosse executada com sucesso. Com o ficheiro XML devidamente configurado, e depois de se ter executado o *deploy* dos *Web Services*, foi desenvolvido um pequeno ficheiro Java que se ligasse, via *Web Services*, à base de dados do RPM, com vista a obter uma base para a integração destes serviços na ferramenta de suporte à gestão de projectos - o *Eagle v1.0*.

3.2.3.5 Migração de base de dados NSN

Para tornar possível o uso dos RPM *Web Services*, teria de ser realizada uma actualização da ferramenta de gestão de base de dados (DB2) e da própria base de dados, porque a versão instalada ainda não o permitia.

O processo de actualização da DB2 começou com a execução de um *backup* da base de dados. Por motivos administrativos, a actualização da DB2 teria de ser realizada noutro computador, então, esse ficheiro de *backup* foi transferido para outra máquina e a DB2 versão 8 foi instalada nessa mesma máquina com sucesso. Neste momento, com o ficheiro de *backup* e com a DB2 versão 8 na mesma máquina, o *restore* desse mesmo *backup* reunia todas as condições para ser executado. Um *script* foi construído de raiz porque este restore teria de mudar as *paths* das *tablespaces* da base de dados, devido à transferência do ficheiro de *backup* para outra máquina. O script foi executado e passado uma hora, o *restore* tinha sido efectuado com sucesso. Assim, já se poderia usar a DB2 versão 8 para fazer a gestão da base de dados, faltando apenas o segundo passo (migração da base de dados da versão 6.1.2.7 para a versão 7.0.1.1) para que se atingisse o objectivo de aceder à base de dados via *Web Services*. Para se executar esta migração outro *script* foi construído. Não foi construído de raiz, já que a IBM disponibilizou-o, mas teve de sofrer algumas alterações. Após ser executado o *script* da migração com sucesso, ficando com a DB2 e a base de dados nas versões correctas, passou-se à fase de testes de utilização dos serviços disponibilizados pelo RPM.

3.2.3.6 Recolha de Dados e Apresentação

Após a ligação com a base de dados do RPM, através da chamada (num ficheiro Java) aos serviços disponibilizados pelos RPM *Web Services*, ter sido comprovada, foi realizada uma reunião com os responsáveis pelo *Eagle v1.0*. Esta reunião teve como objectivo obter uma percepção do que seria necessário para a integração do ficheiro Java desenvolvido. Chegou-se à conclusão que teria de se criar um *bean* que tratasse da chamada aos RPM *Web Services* e se pudesse atingir o objectivo final - a construção de gráficos que permitissem a monitorização de tarefas de projectos de *Network Management*, a partir de dados conseguidos através de *Web Services*.

Apesar da documentação sobre os RPM *Web Services* ser escassa, os dados pretendidos para realizar os gráficos foram recolhidos. No ficheiro Java (*bean*) o primeiro passo realizado foi aceder aos RPM *Web Services* (previamente instalados no *application server*) fazendo o *login*, da seguinte forma:

```
authenticate = authlocator.getAuthenticate(new URL(url + "Authenticate"));
sessionid = authenticate.login(username, password, dsn);
System.out.println("Login: SESSIONID -> "+ sessionid);
applicationInterface = applicationLocator.getApplication(new URL(url + "Application"));
```

Figura 3.16: *Web Services Login*

Se o *login* (*username* e *password*) no RPM (localização dos *Web Services* representado pela variável *url*), for realizado com sucesso um *id* de sessão é retornado. Este *id* de sessão contém informação usada pelos *Web Services* para identificar o utilizador cliente através das transacções sem ter que enviar o *username* e *password* cada vez que se faça uma operação. O *Web Service* atribui um *id* de sessão ao cliente no *login*, que é válido até o cliente enviar um pedido de *logout* para o serviço *Authenticate* para terminar a sessão. Com o *login* efectuado com sucesso, o próximo passo consistia em recolher dados sobre determinados projectos (excerto de código apresentado na figura 3.17):

```
String xpath = "/Project[@name='BPR V1.0']/Task/ResourceTaskAssignment/TaskAssignment";
load = applicationInterface.loadFromXpath(sessionid, xpath, null);

if(load.getRpmObjectList()==null || load.getRpmObjectList().length==0)
    System.out.println("Project not found!");

else{
    System.out.println("Project found!");
    RPMObject [] projects = load.getRpmObjectList();
```

Figura 3.17: *Recolha de dados através de Web Services*

A recolha de dados é realizada através de uma função de *load* de uma *XPath*. Esta *XPath*, na interface do *Web Service*, é usada para obter dados da base de dados

do RPM. Neste caso, a informação que se quer obter é tudo o que diga respeito às tarefas associadas ao projecto *BPR v1.0*, de modo a perceber se essas tarefas estão, ou não, atrasadas em relação ao inicialmente planeado.

Depois dos dados recolhidos terem sido processados, com o ficheiro XML previamente configurado (configuração do *bean*), e com um ficheiro JSP definindo o tipo de gráfico (usando a tecnologia Jenia) que iria representar esse *bean*, o objectivo final foi cumprido.

3.2.4 Resultados

Após a aprendizagem de inúmeras tecnologias, foi possível atingir, com sucesso, o objectivo deste projecto - apresentar graficamente dados recolhidos de um determinado projecto através de *Web Services*, integrado numa ferramenta de suporte à gestão. Três tipos de gráficos foram desenvolvidos:

- Esforço actual - Gráfico que apresenta o esforço actual de cada tarefa do projecto a gerir;
- Esforço esperado - Gráfico que apresenta o esforço esperado de cada tarefa até ao fim do projecto;
- Recursos - Gráfico que representa a quantidade de recursos gastos considerando o esforço actual e o esforço esperado de um determinado projecto.

O gráfico de esforço actual está ilustrado na figura 3.18. Como se pode constatar

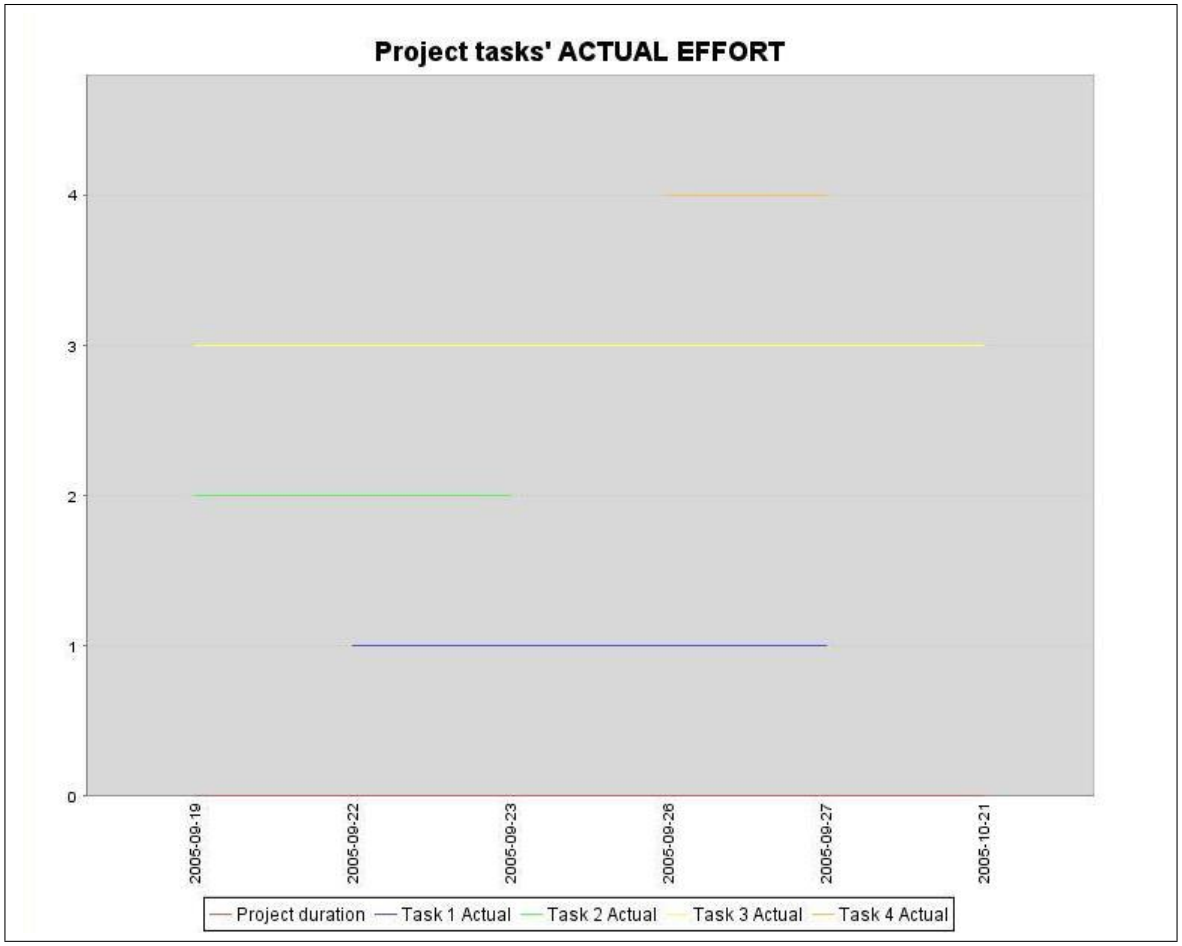


Figura 3.18: Esforço actual do projecto escolhido

no gráfico, o projecto em questão é constituído por quatro tarefas, estando atribuído a cada uma delas, o tempo actual de execução das mesmas. O gráfico apresentado na figura 3.19, representa o esforço esperado de cada tarefa de um projecto. Gráfico semelhante ao primeiro, com a diferença de que os dados apresen-

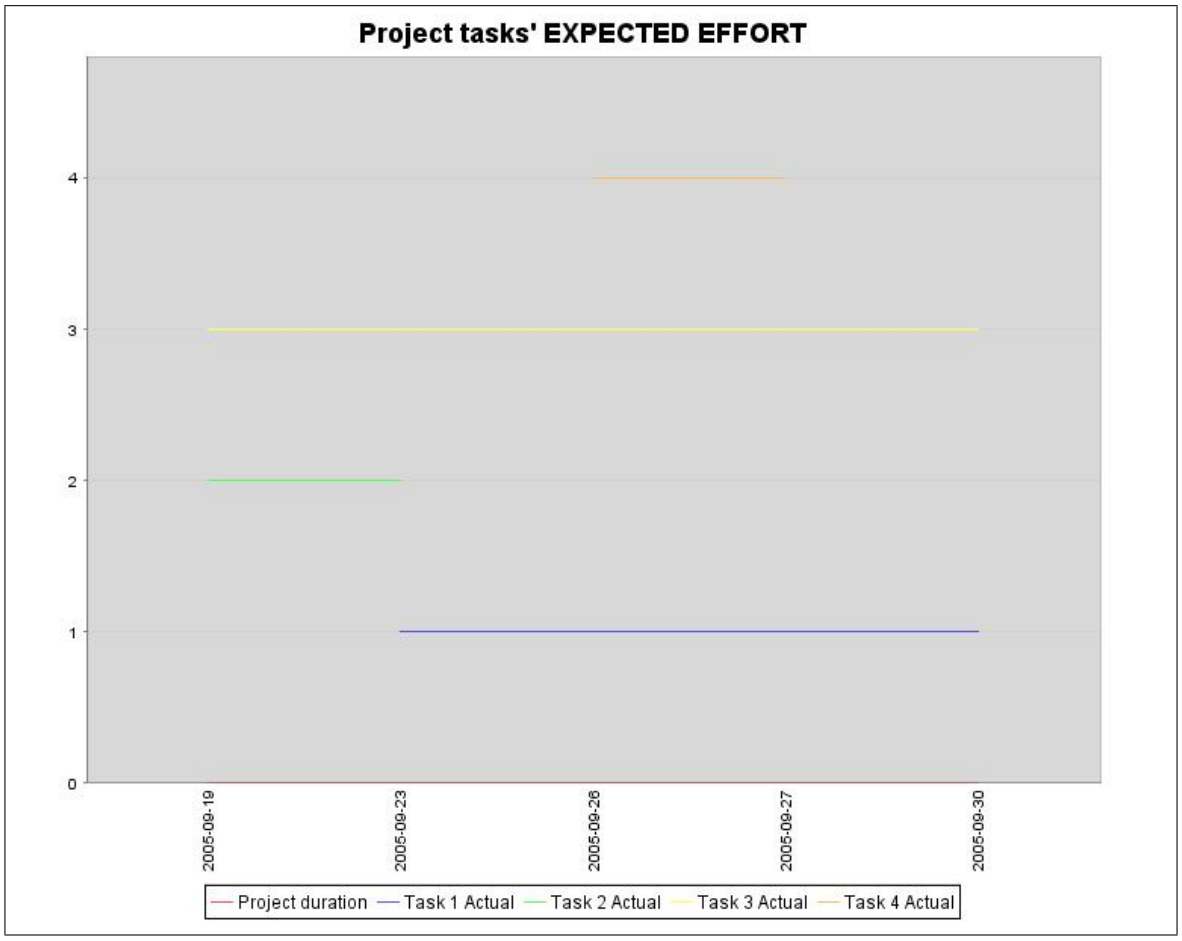


Figura 3.19: Esforço esperado do projecto escolhido

tados dizem respeito ao tempo esperado de execução de cada tarefa pertencente ao projecto.

Por último é apresentado, na figura 3.20, o gráfico que representa a quantidade de recursos, segundo o esforço actual e esperado, alocados ao projecto. Ao analisar o gráfico pode-se concluir que, os recursos usados actualmente não seriam os esperados. Ao olhar para o gráfico, por exemplo, no dia 22-09-2005 deviam estar, idealmente, apenas duas pessoas a trabalhar neste projecto em vez de três.

Esta nova funcionalidade veio introduzir uma maior facilidade, para os gestores de projectos e não só, em visualizar de uma forma mais perceptível como decorre o projecto de acordo com o esperado inicialmente.

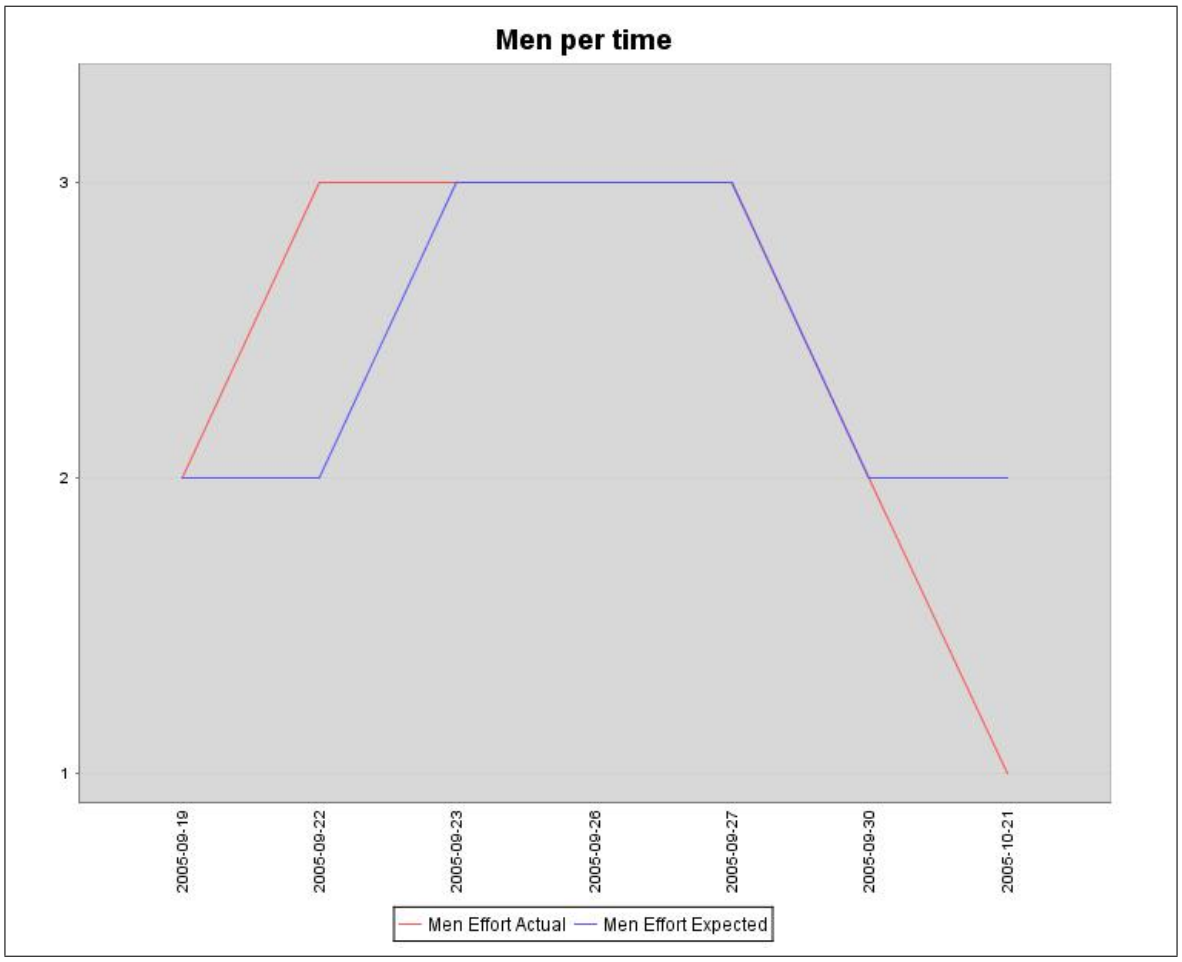


Figura 3.20: Recursos

3.3 Resultados Globais

Os resultados globais do estágio foram muito positivos, mesmo que o primeiro projecto não tenha sido realizado na sua totalidade (por razões alheias à equipa UXG). Todas as tarefas intermédias foram desenvolvidas com sucesso, tais como o levantamento de requisitos, a análise de CMSs e motores de busca a integrar no portal *web* e a construção de um protótipo, tendo sido construído com base nas *guidelines* produzidas (de acordo com as regras da antiga Siemens S.A.). Após a paragem do primeiro projecto (Ferramentas de Suporte à Gestão de Conteúdos), foi proposto um segundo projecto (Suporte à Gestão de Projectos) em que o responsável por este último projecto fazia parte de outra equipa. À medida que se ia desenvolvendo o projecto, a comunicação, tanto com o responsável deste projecto, como com os restantes membros da equipa era constante. Devido ao esforço do estagiário na aprendizagem de novas tecnologias, o bom relacionamento entre as duas equipas e o espírito de entreajuda, o projecto Suporte à Gestão de Projectos foi concluído com sucesso - acedeu-se à base de dados da NSN através de *Web Services*, apresentando

graficamente dados (esforço actual, esforço esperado e recursos) sobre determinados projectos, com o objectivo de facilitar a visualização dos mesmos.

A interacção com os responsáveis dos diferentes projectos através de reuniões foi muito produtivo para o crescimento pessoal e profissional do estagiário. A aprendizagem de inúmeras tecnologias (tanto no primeiro projecto como no segundo) que desconhecia, ou que eram pouco familiares, foi um ponto muito positivo. Permitiu por um lado, solidificar os conhecimentos que o estagiário já tinha adquirido na faculdade e por outro lado, aprender novas tecnologias de modo a aumentar o conhecimento.

Capítulo 4

Conclusão e Trabalho Futuro

Neste estágio foi desenvolvido um trabalho de análise, investigação e de implementação que permitiu facilitar e optimizar a gestão de informação, seja esta informação sobre conteúdos, equipas, tarefas ou projectos.

No primeiro trabalho, focado na gestão de conteúdos, foi efectuado um levantamento de requisitos junto dos responsáveis do projecto, permitindo à equipa perceber quais os que seriam mandatórios e opcionais. Com os requisitos bem definidos, concluiu-se que a integração de um CMS neste projecto, que disponibilizasse grande parte desses requisitos, seria muito útil. De entre os vários analisados, o CMS Alfresco foi o proposto. Adicionalmente, diversos motores de busca foram analisados para integrar no projecto, tendo sido descoberta a melhor opção para propor - DocSearcher. Um protótipo do portal *web* foi construído a partir das *guidelines* produzidas, após várias reuniões para definir o *layout* e a organização da informação.

Como trabalho futuro, caso o projecto tenha continuidade, fica a integração do CMS Alfresco, do motor de busca DocSearcher e do protótipo baseado no conceito WYSIWYG (poderá resultar numa patente NSN) no portal *web* desenvolvido, bem como a concretização de todos os requisitos mandatórios já descritos.

Para permitir adicionar as novas funcionalidades, de monitorização de projectos e recursos, ao projecto *Eagle*, concluiu-se que a utilização de *Web Services*, para recolher dados da base de dados do RPM (onde se encontram todos os recursos, tarefas e projectos relacionados com a NSN), é uma solução robusta e fiável. Depois da configuração de todo o ambiente, que passou por actualizar a ferramenta que gere a base de dados, o próprio repositório e a instalação dos *Web Services* do RPM, implementou-se no *Eagle* novas funcionalidades. Funcionalidades essas que permitiram gerar, automaticamente, gráficos com a duração actual e esperada de todas as tarefas associadas a um projecto, bem como, apresentar (graficamente) os recursos gastos se o projecto continuasse com o esforço actual, ou se gastasse o esforço esperado antes do início do mesmo. Estes gráficos permitem visualizar, com maior facilidade, se o projecto, com os recursos actuais, está a decorrer dentro do

esperado e se está a utilizar os recursos previamente estabelecidos. Estes dados que permitiram a execução dos gráficos foram recolhidos, na sua totalidade, via *Web Services*.

Para trabalho futuro, a implementação destes gráficos deveria ser alargada para todos os projectos abrangidos pela ferramenta de suporte à gestão utilizada, o *Eagle*, usando sempre, para a recolha de dados, os *Web Services*. Outra funcionalidade interessante seria a visualização gráfica de *error forecasts*, que permitisse prever a data de conclusão do projecto, consoante os erros ocorridos nos casos de teste.

No global, com esta aprendizagem contínua, o estagiário passou a dominar as principais tecnologias com as que está a trabalhar neste momento:

- linguagem de programação Java;
- EJBs;
- XML;
- *Web Services*;
- IBM RPM;
- IBM DB2;
- Jenia.

Outro aspecto positivo foi a aprendizagem de novas capacidades de trabalho, tais como: realizar o levantamento de requisitos, interagir com pessoas responsáveis por projectos (discutindo ideias e planeando o trabalho a realizar), flexibilidade na aprendizagem de inúmeras tecnologias, capacidade em realizar diferentes tipos de tarefa e essencialmente, trabalho de investigação que permitiu ultrapassar obstáculos de grande dificuldade durante o estágio.

Este período vivido na Siemens S.A., e mais recentemente na NSN, permitiu ao estagiário aumentar a sua experiência, tanto a nível pessoal como a nível profissional. A flexibilidade para aprender e para trabalhar com todas as tecnologias apresentadas, foi a característica mais importante presente no decorrer do estágio.

Acrónimos

APAC	Asian Pacific
API	Application Programming Interface
ASP	Active Server Pages
BMP	Bean Managed Persistence
BPR	Business Process Re-engineering
CHM	Microsoft Compiled HTML Help
CMP	Container Managed Persistence
CMS	Content Management System
CSS	Cascade Style Sheet
DB	Database
DHTML	Dynamic Hyper Text Markup Language
DOC	Word Document
EIS	Enterprise Information System
EJB	Enterprise Java Beans
FUM	Follow Up Meeting
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IC	Information and Communications
JAR	Java Archive
JAX-RPC	Java API for XML-based Remote Procedure Call)
JAX-WS	Java API for XML-based Web Services
JDBC	Java Database Connectivity
JEE5	Java Enterprise Edition 5
JSP	JavaServer Pages

NSN	Nokia Siemens Networks
OG	Operating Group
PDF	Portable Document Format
PHP	Hypertext Preprocessor
PPT	PowerPoint
RAR	Roshal Archive
RPM	Rational Portfolio Manager
RTF	Rich Text Format
SI	Sistemas de Informação
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
TXT	Text format
UI	User interface
URL	Uniform Resource Locator
UXG	User eXperience Group
W3C	World Wide Web Consortium
WSDL	Web Services Description Language
WYSIWYG	What You See Is What You Get
XHTML	eXtensible Hypertext Markup Language
XLS	Excel
XML	Extensible Markup Language
XSL	Extensible Style Language
ZIP	Zone Information Protocol

Bibliografia

[1] Feliz Ribeiro Gouveia. Gestão da informação. Technical report, Universidade Fernando Pessoa, 2000.

[2] Chris Peltz. Web services orchestration, a review of emerging technologies, tools, and standards. Technical report, Hewlett Packard Co., 2003.

[3] W3C. Html 4.01 specification. Disponível em http://www.w3.org/TR/html4/, acedido em Outubro 2006.

[4] JavaScript. Javascript tutorial. Disponível em http://www.w3schools.com/js, acedido em Outubro 2006.

[5] W3C. Cascading style sheets. Disponível em http://www.w3.org/Style/CSS/, acedido em Outubro 2006.

[6] Microsoft. Visual basic developer center. Disponível em http://msdn2.microsoft.com/en-us/vbasic, acedido em Outubro 2006.

[7] Sun. Java technology. Disponível em http://java.sun.com/, acedido em Outubro 2006.

[8] Sun. Javaserver faces technology. Disponível em http://java.sun.com/javaee/javaserverfaces, acedido em Abril 2007.

[9] AJAX. Ajax tutorial. Disponível em www.w3schools.com/ajax, acedido em Maio 2007.

[10] Apache. Welcome to lucene. Disponível em http://lucene.apache.org, acedido em Fevereiro 2007.

[11] Alfresco. Alfresco - open source enterprise content management including web content management. Disponível em http://www.alfresco.com/, acedido em Novembro 2006.

[12] DotNetNuke. Dotnetnuke is a free, open source framework ideal for creating enterprise web applications. Disponível em http://www.dotnetnuke.com/, acedido em Novembro 2006.

[13] W3C. The w3c markup validation service. Disponível em http://validator.w3.org, acedido em Dezembro 2006.

[14] DocSearcher. Docsearcher is a search tool. Disponível em http://docsearcher.henschelsoft.de, acedido em Fevereiro 2007.

[15] Regain. Regain - your hidden information. Disponível em http://regain.sourceforge.net, acedido em Fevereiro 2007.

[16] Zilverline. The reverse search engine. Disponível em http://www.zilverline.org, acedido em Fevereiro 2007.

[17] QFS Quality First Software. Qf-test version 2.0. Disponível em http://www.qfs.de/en/qftestJUI, acedido em Novembro 2006.

[18] Mercury. Regression testing - mercury winrunner. Disponível em http://www.mercury.com/us/products/quality-center/functional-testing/winrunner/, acedido em Novembro 2006.

[19] IBM. Ibm - rational portfolio manager - rational portfolio manager - software. Disponível em http://www-306.ibm.com/software/awdtools/portfolio, acedido em Março 2007.

[20] IBM. Ibm - ibm rational clearcase v7 - rational clearcase - software. Disponível em http://www-306.ibm.com/software/awdtools/clearcase, acedido em Março 2007.

[21] Sun. Javaserver pages technology. Disponível em http://java.sun.com/products/jsp/, acedido em Abril 2007.

[22] W3C. Extensible markup language. Disponível em http://www.w3.org/XML/, acedido em Janeiro 2007.

[23] W3C. Soap version 1.2 part1: Messaging framework (second edition). Disponível em http://www.w3.org/TR/soap12-part1, acedido em Fevereiro 2007.

[24] Sun. Enterprise javabeans technology. Disponível em http://java.sun.com/products/ejb/, acedido em Abril 2007.

[25] Apache. The apache ant project. Disponível em http://ant.apache.org, acedido em Janeiro 2007.

[26] Apache. Apache tomcat. Disponível em http://tomcat.apache.org, acedido em Março 2007.

[27] Apache. Webservices - axis. Disponível em http://ws.apache.org/axis/, acedido em Março 2007.

[28] SQL. Sql tutorial. Disponível em http://www.w3schools.com/sql/, acedido em Março 2007.

[29] IBM. Db2 universal database. Disponível em http://publib.boulder.ibm.com/infocenter/db2luw/v8, acedido em Março 2007.

[30] Jenia. jenia.org. Disponível em http://www.jenia.org/, acedido em Abril 2007.

[31] Java. jax-rpc: Jax-rpc reference implementation. Disponível em https://jax-rpc.dev.java.net/, acedido em Fevereiro 2007.

[32] Java. jax-ws: Jax-ws reference implementation. Disponível em https://jax-ws.dev.java.net/, acedido em Março 2007.

[33] Eclipse. Eclipse - an open development platform. Disponível em http://www.eclipse.org/, acedido em Janeiro 2007.

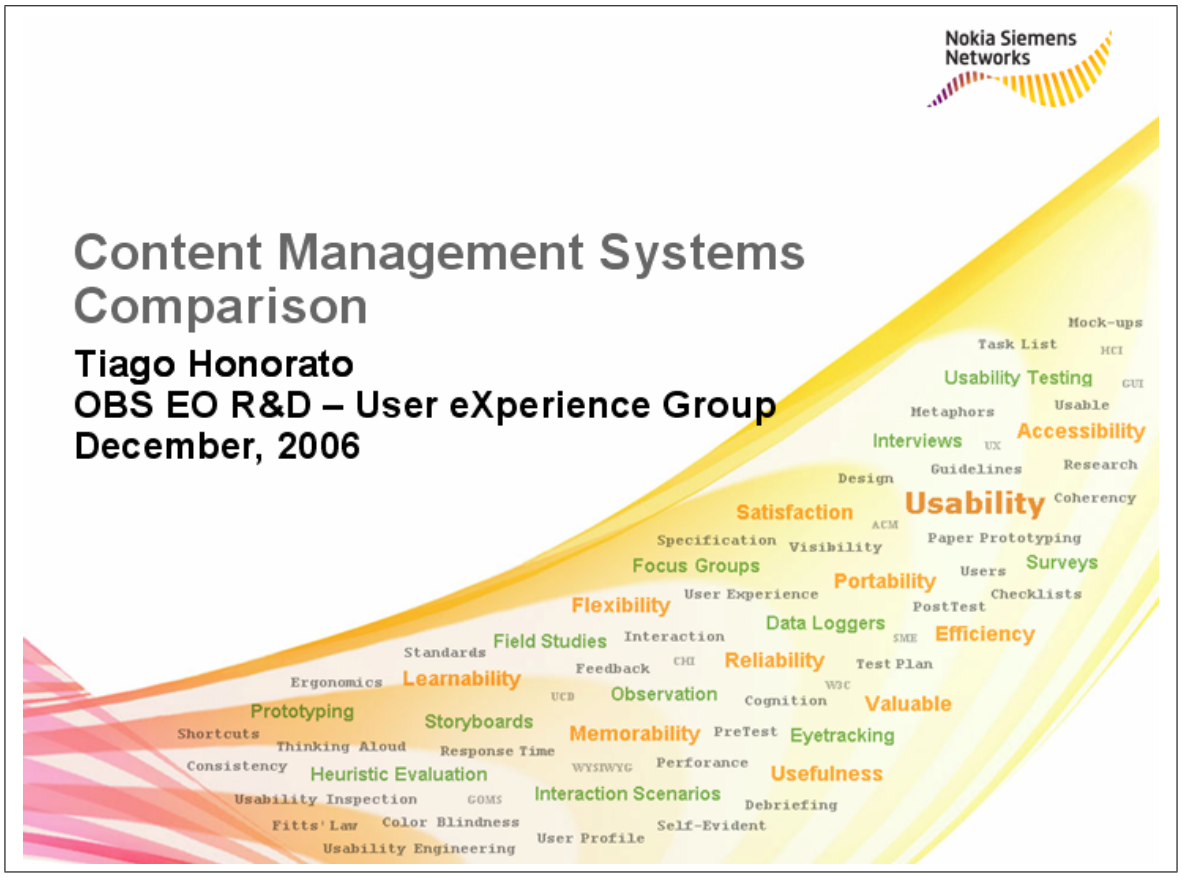
[34] W3C. Web services description language (wsdl) 1.1. Disponível em http://www.w3.org/TR/wsdl, acedido em Fevereiro 2007.

[35] Hibernate. Hibernate.org. Disponível em www.hibernate.org, acedido em Março 2007.

[36] Oracle. Oracle 10g, siebel, peoplesoft | oracle, the world's largest enterprise software company. Disponível em http://www.oracle.com, acedido em Março 2007.

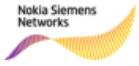
Apêndice A

Content Management Systems Comparison



Scope

- The main goal - choose a CMS that can provide all the mandatory requirements
- How?
  - Gathering all CMS' requirements of all existing CMSs
  - Applying filters in order to get the best CMS
- Two Open Source CMSs will be presented in detail – DotNetNuke and Alfresco
- Which one is better? DotNetNuke or Alfresco?



Collecting all CMSs

- With the help of www.cmsmatrix.org all, or most, of the CMSs were collected (total: 645)
- Then, twelve filters were applied by a specific order - requirements' importance
- The final result, of the filters' application, were four CMSs:
  - Alfresco
  - AxCMS.net 6
  - DotNetNuke
  - OpenEdit
- Between these four CMSs two were chosen to test in detail:
  - DotNetNuke and Alfresco

Requirements

- The most important requirements and its description:

WYSIWYG Editor	A web-based rich text editor to allow publishers to create formatted content without knowing HTML, CSS, XML, XSL
Application Programming Interface (API)	Possibility to change or extend functionalities
Granular Privileges	Read and write privileges per page, content or functionality
Template management	Styles and templates' management to define design and layout
Workflow	The automation of a business process management - A process description of how tasks are done, by whom, in what order and how quickly.
Search Engine	Can index the managed content and allow the user to search the indexed content

DotNetNuke

- DotNetNuke is an open source CMS, ideal for creating web applications
- Richard Dudley:
  - *"My thinking is Extranet=DNN. DNN is designed to be used and maintained by persons with little IT experience."*
- Website:
  - www.dotnetnuke.com
- DotNetNuke's last update was in 30/11/2006 and it has an active community (forum and blog)
- System Requirements:
  - Programming Language – ASP.NET
  - Application Server – Internet Information Services (IIS)
  - Database – MSSQL 2000/2005, MSSQL Express 2005
  - Operating System - Windows

DotNetNuke (Continued)

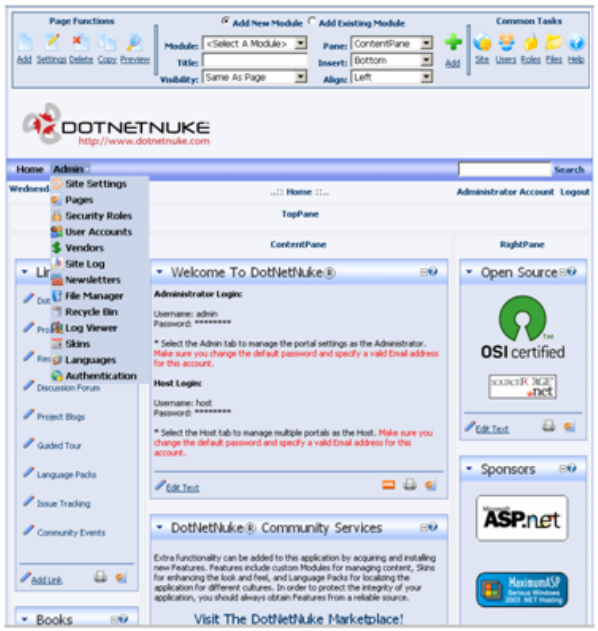
- Advantages:
  - Open source – allows developers to modify or extend the source code according to their needs
  - Superb community support
  - Easy website creation
  - Supports multiple child portals
  - Can be extended with free modules (positioning, personalization, minimize capabilities, etc.)
  - Can serve as an e-commerce platform, host photo galleries, etc.
  - Good skinning engine and lots of skins available
  - Drag and drop components in the website

DotNetNuke (Continued)

- Disadvantages:
  - Application Programming Interface (API) not properly documented
  - Complicated documents explaining how to add modules
  - Too many time is spent adding a new feature to DotNetNuke code base
  - There isn't a tight integration with Office
  - Doesn't have document library with version control
  - Search inside the documents in its libraries is not possible
  - Stores documents as files – documents can be downloaded directly and that leads to a security problem

DotNetNuke (Continued)

- Screenshot:



Nokia Siemens Networks

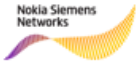
Alfresco

- Alfresco is an open source CMS that's more document-focused, instead of web applications' creation, like DotNetNuke
- Website:
  - www.alfresco.com
- Afresco's last update was in 6/10/2006 and it has an active community (forum, blog and Wiki)
- System Requirements:
  - Programming Language – Java
  - Application Server – Any that supports Java 1.5
  - Database – Hibernate-supported databases
  - Operating System - Any

Nokia Siemens Networks

Alfresco (Continued)

- Advantages:
  - Open source
  - API well defined and provide JavaDocs for developers
  - It has a powerfull content rules framework
  - Good content versioning system
  - Easy to use and simple user interface
  - Robust workflow
  - Advanced search function
  - Introduces discussions
  - Good community support
  - Integration with PHP and .NET applications
  - Permit enterprises to extend Alfresco and integrate it with other systems



Alfresco (Continued)

- Disadvantages:
  - Alfresco .java source files not provided
  - Lacks the capability of publishing websites
  - Non-free features
  - Can't drag and drop components in website
  - Limited sitemap
  - Skins not available
  - It hasn't friendly URLs
    - Example – When a space is created, the URL, instead of contain the space's name, presents the space ID

Alfresco (Continued)

- Screenshot:

Conclusion

- After presenting the main advantages/disadvantages of DotNetNuke and Alfresco, here are the conclusions:
  - DotNetNuke is focused in websites creation and Alfresco is focused in documents' management (similar to Microsoft SharePoint)
  - Extranet = DotNetNuke; Intranet = Alfresco
  - DotNetNuke or Alfresco?
    - Each one has its job... "*choose the right tool for the right job*"
  - Choose DotNetNuke if:
    - The purpose of the project is to create external websites, where standard message boards, blog, calendar, forum and files are required
  - Choose Alfresco if:
    - The most important requirements of the project are managing documents, providing documents approval and workflow

References

- The CMS Matrix – cmsmatrix.org – The Content Management Comparison Tool, http://www.cmsmatrix.org/, accessed December 6, 2006
- DotNetNuke > Home (DNN 4.3.7), 2002-2006, http://www.dotnetnuke.com, accessed December 6, 2006
- Richard Dudley: SharePoint or DotNetNuke?, 2006, http://aspadvice.com/blogs/rjdudley/archive/2006/01/18/14755.aspx, accessed December 6, 2006
- Fear and Loathing: DotNetNuke vs. SharePoint, the big showdown, 2006, http://weblogs.asp.net/bsimser/archive/2006/01/31/437023.aspx, accessed December 6, 2006

References (Continued)

- Alfresco – Open Source Enterprise Content Management (CMS) including Web Content Management, 2005-2006, http://www.alfresco.com/, accessed 6 December, 2006
- KMWorld.com: Alfresco tries to repeat history, 2005, http://www.kmworld.com/Articles/ReadArticle.aspx?ArticleID=14544, accessed 6 December, 2006
- Open Sources | InfoWorld | Alfresco: The SharePoint killer?, 2005, http://weblog.infoworld.com/openresource/archives/2005/10/alfresco_the_sh.html, accessed 6 December, 2006
- Alfresco Enterprise Content Management 1.0 Released, 2006, http://www.theserverside.com/news/thread.tss?thread_id=37384, accessed 6 December, 2006

Contacts

Tiago Honorato
OBS EO R&D / Lisbon / Portugal
tiago.honorato@siemens.com

Nokia Siemens Networks

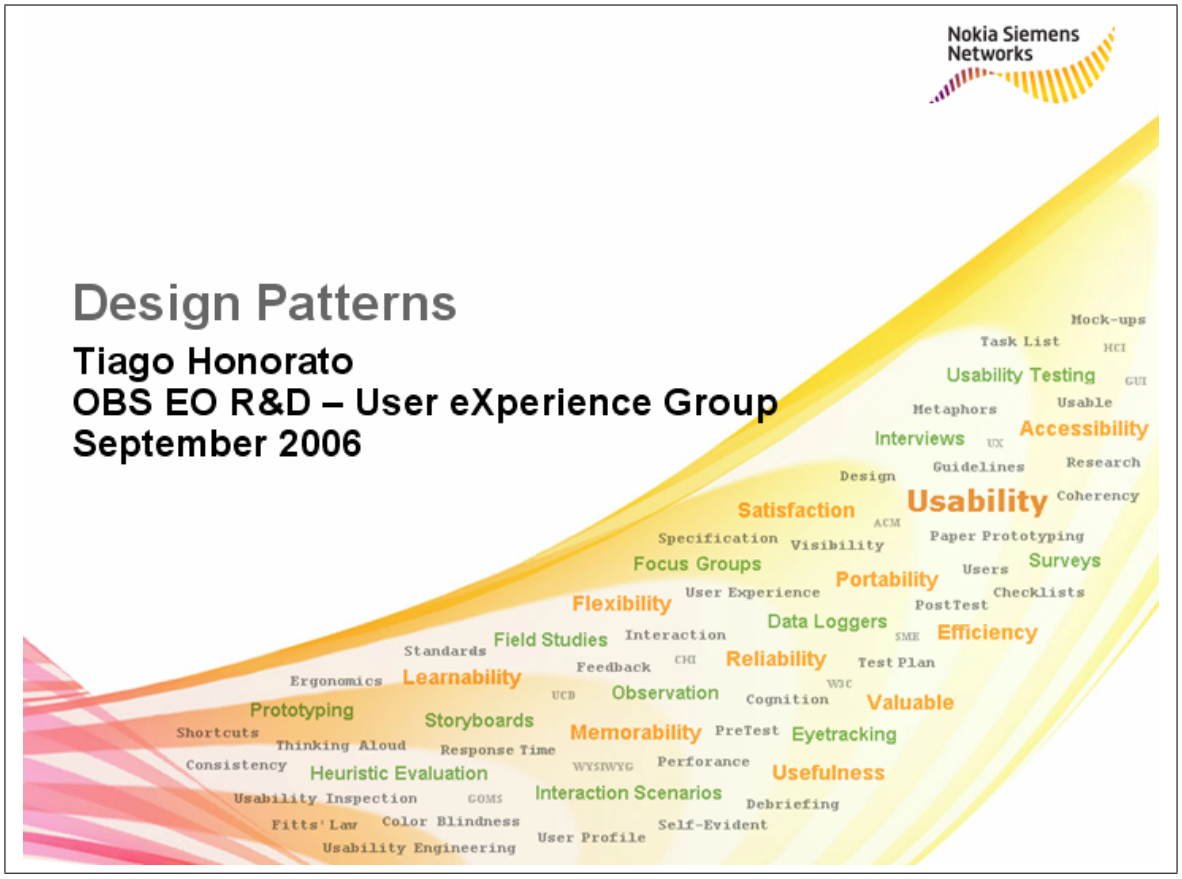
User eXperience Group

Nokia Siemens Networks Portugal, S.A.
Rua Irmãos Siemens, N.º1
2720-093 Alfragide
Fax: 351 – 21 416 7502

Nokia Siemens Networks

Apêndice B

Design Patterns



Scope

- Review the Unified Modeling Language (UML)
- Design Patterns are defined in this language
- The goal is to ensure that on the one hand, all Design Patterns are presented, and on the other hand, well explained, using UML
- In the end of the presentation the viewers, hopefully, will get a general idea of all existing design patterns

Nokia Siemens Networks

UML

Design Patterns using UML class diagrams:

- Show static class relationships

Design Patterns using UML Sequence diagrams:

- Show dynamic object interaction

Objects:

- "An *object* has a *state*, *behavior* and *identity*…", from Grady Booch
- State = data
- Behavior = operations that change or query the object's state
- Identity = an object must be unique and strictly defined

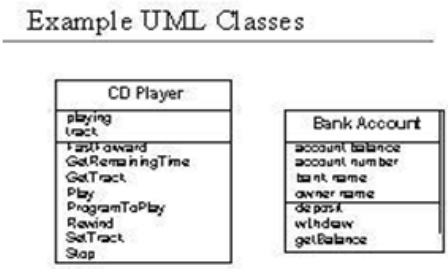
Nokia Siemens Networks

UML (continued)

Classes:

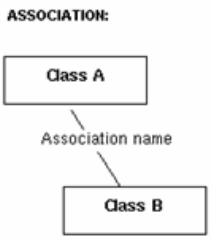
- Similar objects are grouped into *classes*
- All objects share identical behavior
- All objects have the same data members
- Inheritance allows classes to have common properties and operations

- Three types of static relationships:
  - Association
  - Aggregation
  - Inheritance



UML (continued)

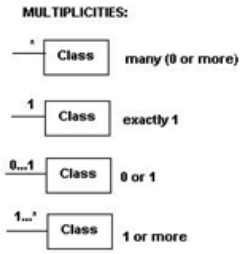
UML Relationships:



Association:

- Represents the association between objects, usually the association "has"

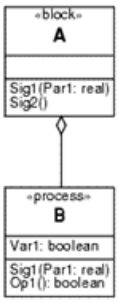
- The associations have multiplicities to show how many participants exists



UML (continued)

Aggregation:

- Represents the object relation “is composed of”, as demonstrated in the following figure:

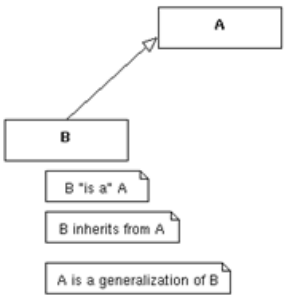


- In the figure, block A contains block B, or block B is part of block A

UML (continued)

Inheritance:

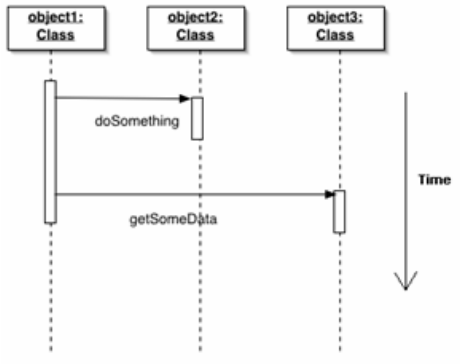
- Represents the object relation ”is a kind of”, as demonstrated in the following figure:



- The bad use of inheritance can lead to design and maintainability problems

UML (continued)

Sequence Diagram (Example):

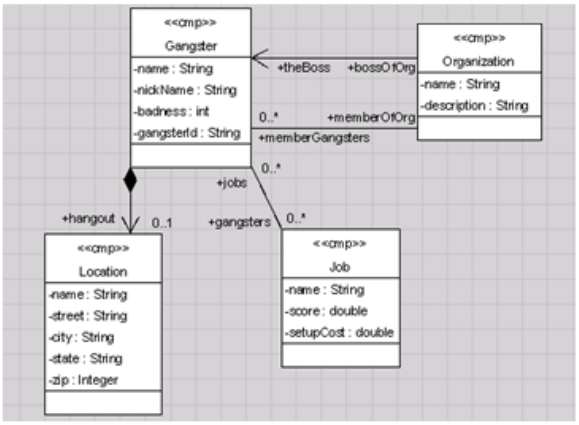


- Object1 interacts with Object2 through the method doSometing(), and interacts with Object3 too, getting necessary data, through the method getSomeData()

Nokia Siemens Networks

UML

Class Diagram (Example):



Nokia Siemens Networks

Description of Design Patterns

A design pattern is a solution to solve frequent design problems

It has four elements:

- Name – For communication & design documentation
- Problem – When to apply the pattern
- Solution – The elements and their relationships
- Consequences – What are the costs or benefits

Nokia Siemens Networks

Classes of Design Patterns

1. General Design Patterns:

- 1.1 – Delegation
- 1.2 – Template Method
- 1.3 – Iterator
- 1.4 – Singleton

2. Interface Design Patterns:

- 2.1 – Façade
- 2.2 – Proxy
- 2.3 – Adapter
- 2.4 – Bridge
- 2.5 – Composite

Nokia Siemens Networks

Classes of Design Patterns (continued)

3. Views and controllers Design Patterns

- 3.1 – Observer
- 3.2 – Mediator
- 3.3 – Command
- 3.4 – Chain of Responsibility
- 3.5 – Visitor

4. Object Creation and Persistence Design Patterns

- 4.1 – Prototype
- 4.2 – Builder
- 4.3 – Factory Method
- 4.4 – Abstract Factory
- 4.5 – Memento
- 4.6 – Flyweight

Classes of Design Patterns (continued)

5. Syntax and Input Analysis Design Patterns

- 5.1 – State Machine
- 5.2 – Strategy
- 5.3 – Interpreter

6. Roles

- 6.1 – State
- 6.2 – Decorator

Classes of Design Patterns (continued)

1. General Design Patterns:

- Single patterns that often occur within other patterns

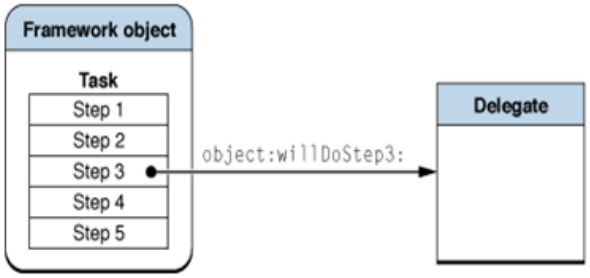
  1.1 – Delegation
  1.2 – Template Method
  1.3 – Iterator
  1.4 – Singleton

Nokia Siemens Networks

1. General Design Patterns

1.1 – Delegation

- An object forwards messages to other objects
- Name:
  - Delegation
- Problem:
  - The "effect" of inheritance



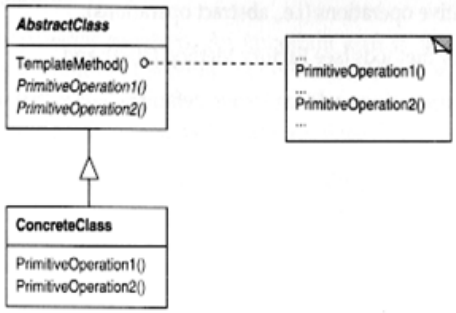
- Solution:
  - Instead of inheritance, use association
- Consequences:
  - Easier adaptation
  - Simplicity and efficiency

Nokia Siemens Networks

General Design Patterns (continued)

1.2 – Template Method

- Define the skeleton of an algorithm, deferring some steps to subclasses
- Name:
  - Template Method
- Problem:
  - All steps of an algorithm in the base class
- Solution:
  - Part(s) of a method deferred to subclass
- Consequences:
  - Assume the form of the function as dictated by the base class



Nokia Siemens Networks

General Design Patterns (continued)

1.3 – Iterator

- Name:
  - Iterator
- Problem:
  - Access to all of the elements in a Collection

```
import java.util.*;
public class Example{
        public static Vector components = new Vector();
        public static void main (String [] args){
                Iterator e = components.iterator();
                while(e.hasNext()){
                        System.out.println("I still have components...");
                        e.next();
                }
        }
}
```

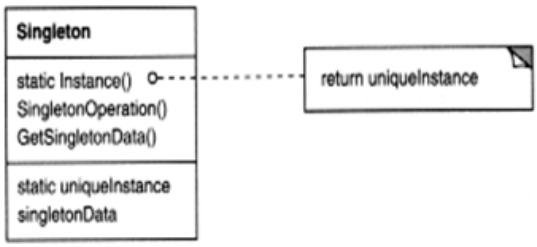
- Solution:
  - Iterator accesses the objects sequentially, in a Collection
- Consequences:
  - Support multiple and simultaneous iterations

Nokia Siemens Networks

General Design Patterns (continued)

1.4 – Singleton

- The Singleton pattern ensures that only one instance of a class is created



- Name:
  - Singleton
- Problem:
  - Ensure an unique instance of a class
- Solution:
  - With a private constructor only the class itself can create instances
- Consequences:
  - Alternative to Singleton – Declare a single instance in the main class – unsafe and undesirable

Nokia Siemens Networks

Classes of Design Patterns (continued)

2. Interface Design Patterns:

- For interfacing software components - networks

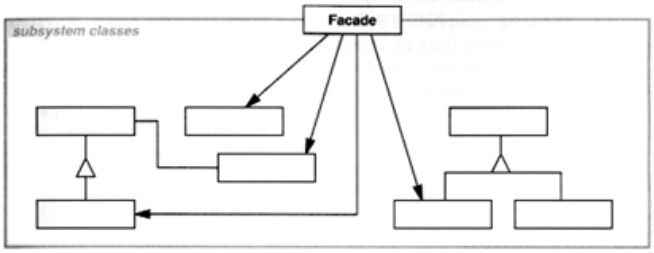
2.1 – Façade
2.2 – Proxy
2.3 – Adapter
2.4 – Bridge
2.5 – Composite

Nokia Siemens Networks

2. Interface Design Patterns

2.1 – Façade

- Interface object for a package of classes



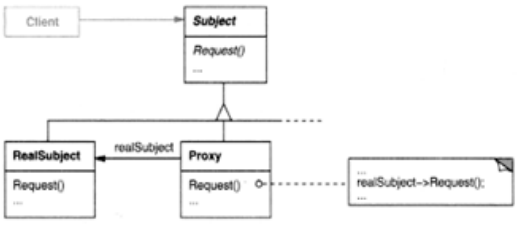
- Name:
  - Façade
- Problem:
  - The work is done by whom?
- Solution:
  - Object Façade delegates the work to other objects
- Consequences:
  - The exposed object is the only member of Façade class
  - Communication with the package's objects

Nokia Siemens Networks

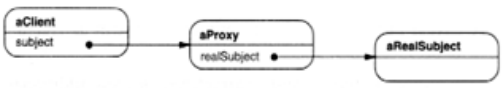
2. Interface Design Patterns (continued)

2.2 – Proxy

- Provides an interface to other object, to control access to it



Here's a possible object diagram of a proxy structure at run-time:



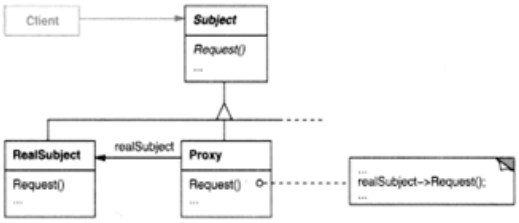
- Name:
  - Proxy
- Problem:
  - Provide location transparency
- Solution:
  - Provide an interface to an object that controls the access to other object
- Consequences:
  - Extra level of indirection in object access

Nokia Siemens Networks

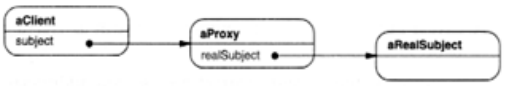
2. Interface Design Patterns (continued)

2.2 – Proxy

- Provides an interface to other object, to control access to it
- Name:
  - Proxy
- Problem:
  - Provide location transparency
- Solution:
  - Provide an interface to an object that controls the access to other object
- Consequences:
  - Extra level of indirection in object access



Here's a possible object diagram of a proxy structure at run-time:



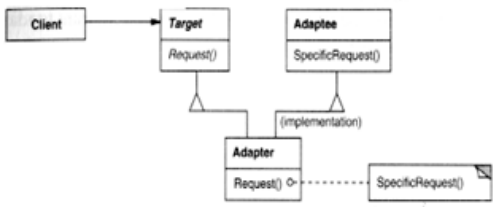
Nokia Siemens Networks

2. Interface Design Patterns (continued)

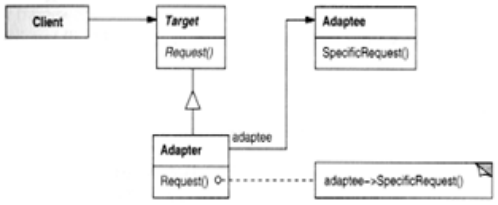
2.3 – Adapter

- Used when a client expects a different interface than that provided
- Name:
  - Adapter
- Problem:
  - Adapt the interface for the client
- Solution:
  - Create a new intermediate
  - It converts the old interface to the new interface
- Consequences:
  - Object adapter lets the adapter work with many adaptees

A class adapter uses multiple inheritance to adapt one interface to another:



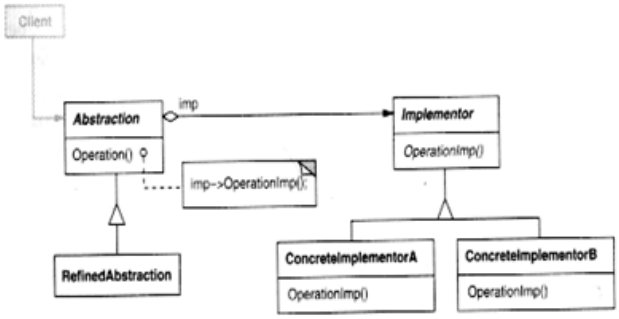
An object adapter relies on object composition:



Nokia Siemens Networks

2. Interface Design Patterns (continued)

2.4 – Bridge

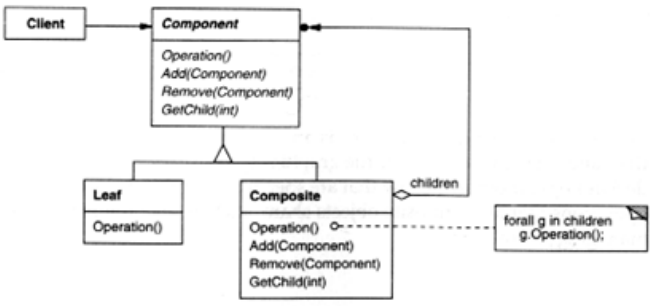


- Separates the abstraction from its implementation
- Name:
  - Bridge
- Problem:
  - Locate dependencies
- Solution:
  - Making a bridge between two inheritance hierarchies (abstractions and implementations)
- Consequences:
  - Abstractions and implementations – independent classes
  - Can be combined dynamically

Nokia Siemens Networks

2. Interface Design Patterns (continued)

2.5 – Composite



- Build complex objects by recursively composing similar objects
- Name:
  - Composite
- Problem:
  - Represent a component that can be atomic or a composite
- Solution:
  - Manipulate objects in a tree
  - Require all the objects in the tree have a common superclass
- Consequences:
  - Useful when implementing tree data structures
  - Several implementation options

Nokia Siemens Networks

Classes of Design Patterns (continued)

3. Views and Controllers Design Patterns:

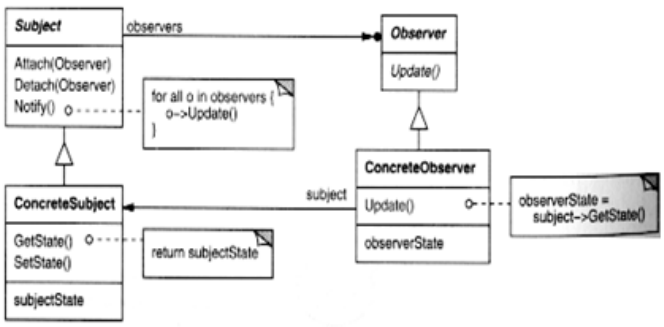
- Used in GUIs and handling dynamic control flow

3.1 – Observer
3.2 – Mediator
3.3 – Command
3.4 – Chain of Responsibility
3.5 – Visitor

Nokia Siemens Networks

3. Views and controllers Design Patterns

3.1 – Observer



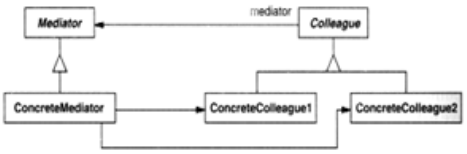
- If an object changes the state, all its dependants are notified and updated about its change
- Name:
  - Observer
- Problem:
  - What to do if the object's state change?
- Solution:
  - Call the notify() method to inform other objects that my state changed
  - The other objects, after that, know they have to update() their state
- Consequences:
  - Implementation and synchronization of observers

Nokia Siemens Networks

3. Views and controllers Design Patterns (continued)

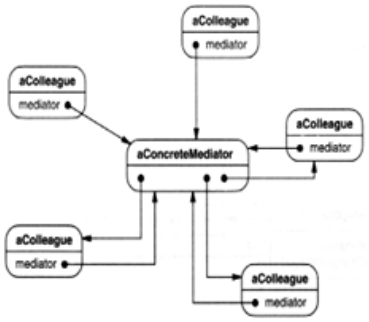
3.2 – Mediator

- Provide an interface object to mediate interaction



- Name:
  - Mediator
- Problem:
  - What to do if one subject's state affects the others?
- Solution:
  - A mediator – manage the state changes of the other objects
- Consequences:
  - Centralize control

A typical object structure might look like this:

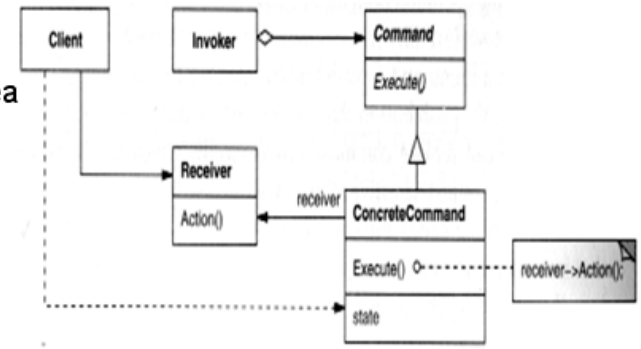


Nokia Siemens Networks

3. Views and controllers Design Patterns (continued)

3.3 – Command

- Saving a document in each edit is not, with sure, a good idea

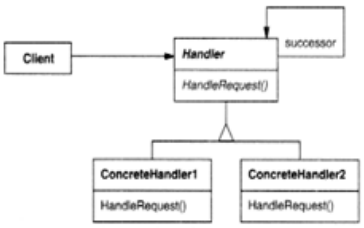


- Name:
  - Command
- Problem:
  - Several edit commands
- Solution:
  - Commands' encapsulation in objects to control or manipulate them
- Consequences:
  - Easy to add new commands

Nokia Siemens Networks

3. Views and controllers Design Patterns (continued)

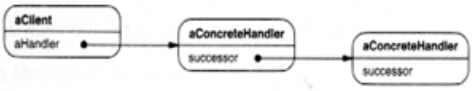
3.4 – Chain of Responsibility



- Send events for a chain of objects in order to try handling them

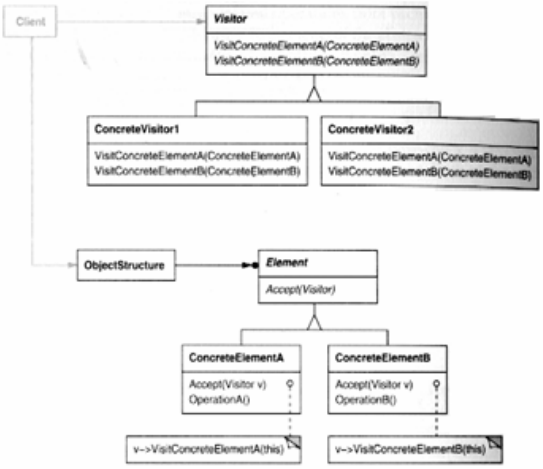
- Name:
  - Chain of Responsibility
- Problem:
  - Who should handle the event?
- Solution:
  - This event is sent to a chain of object
  - Each object can handle the event or pass it to other object
- Consequences:
  - Unknown object(s) will receive it – Delivery is not guaranteed

A typical object structure might look like this:



3. Views and controllers Design Patterns (continued)

3.5 – Visitor



- Apply many operations to objects
- Name: Visitor
- Problem:
  - Add new operations to a structure of objects
- Solution:
  - A visitor – allows putting all data in a different visitor class
  - Add new operation - It isn't necessary to change element objects
- Consequences:
  - Get rid of switch statements

Classes of Design Patterns (continued)

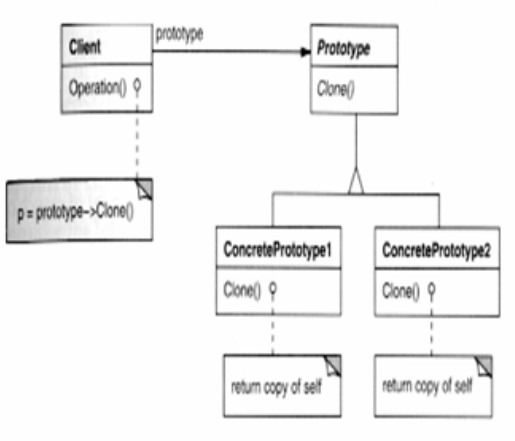
4. Object Creation and Persistence Design Patterns:

- Used in saving/restoring state and constructing objects

4.1 – Prototype
4.2 – Builder
4.3 – Factory Method
4.4 – Abstract Factory
4.5 – Memento
4.6 – Flyweight

4. Object Creation and Persistence Design Patterns

4.1 – Prototype

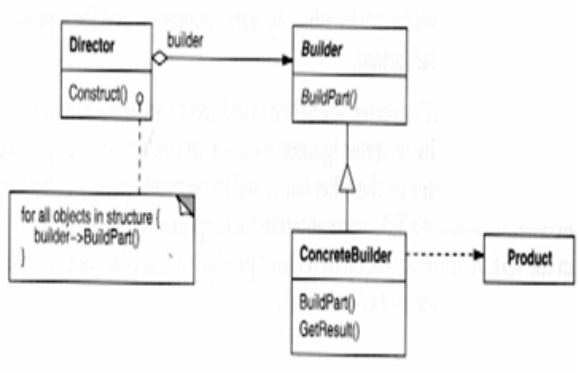


- Create objects from a prototypical instance
- Name: Prototype
- Problem:
  - Cloning object from the class
- Solution:
  - Create a prototype instance
  - Clone that instance to create objects from the same type
- Consequences:
  - Adding new object types – It's easier
- Useful when creating an object - means a significant effort

4. Object Creation and Persistence Design Patterns (continued)

4.2 – Builder

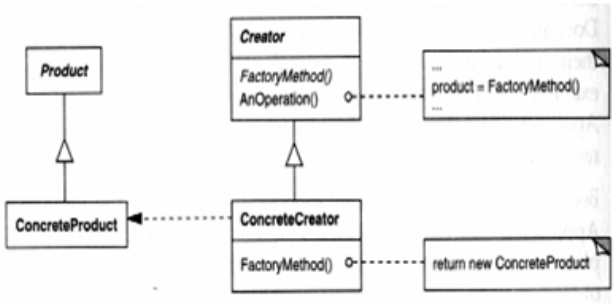
- Allows putting together a class, from parts of that class
- Name: Builder
- Problem:
  - An object has a stack of different classes to create
- Solution:
  - Get all the parts of each class and builder gather them all
- Consequences:
  - Build methods must be declared in the base class
- The client is shielded from the details of the object's construction



4. Object Creation and Persistence Design Patterns (continued)

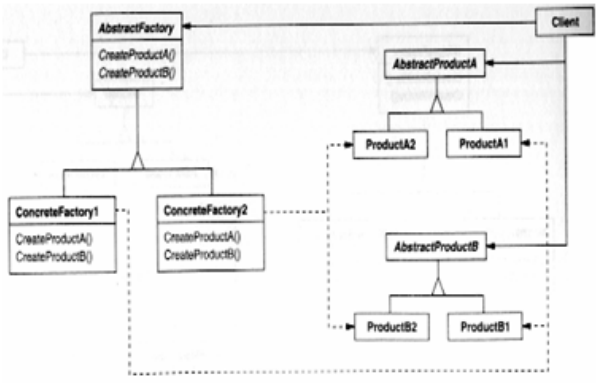
4.3 – Factory Method

- Creation of objects
- Name:
  - Factory Method
- Problem:
  - create objects without specify the exact class of that objects
- Solution:
  - Define a separate method to create objects
  - Subclasses decide which classes will be instantiated
- Consequences:
  - Useful when is needed an object which can assume several forms



4. Object Creation and Persistence Design Patterns (continued)

4.4 – Abstract Factory

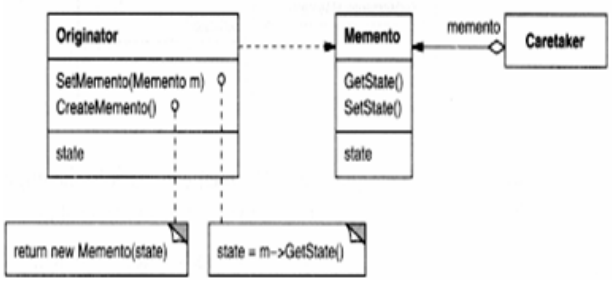


- Create abstract classes' instances, from a group of concrete subclasses
- Name: Abstract Factory
- Problem:
  - Two different classes with the same theme
- Solution:
  - Create an abstract class for the classes with the same theme
- Consequences:
  - Client dynamically chooses a factory
  - Different product types are returned by abstract factory

Nokia Siemens Networks

4. Object Creation and Persistence Design Patterns (continued)

4.5 – Memento



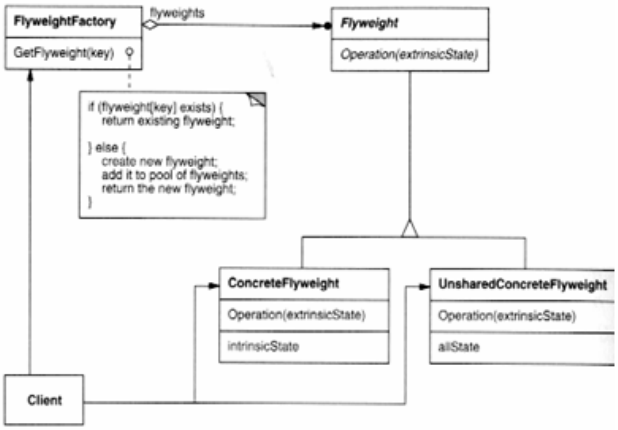
- Like a snapshot of the object's state
- Name:
  - Memento
- Problem:
  - How to implement snapshots?
- Solution:
  - Separate classes that manage data and save the state
- Consequences:
  - Doubt at the time of Memento's creation

- Useful to interact with databases in OO – All objects keep mementos

Nokia Siemens Networks

4. Object Creation and Persistence Design Patterns (continued)

4.6 – Flyweight



- Flyweight pattern allows sharing the state of many objects efficiently
- Name:
  - Flyweight
- Problem:
  - Client object doesn't store shared data
- Solution
  - Call a FlyweightFactory to get a Flyweight
- Consequences:
  - Avoid the expense of multiple instances (containing same info), sharing only one

Nokia Siemens Networks

Classes of Design Patterns (continued)

5. Syntax and Input Analysis Design Patterns:

- Recognizing input data formats and simplifying control flow

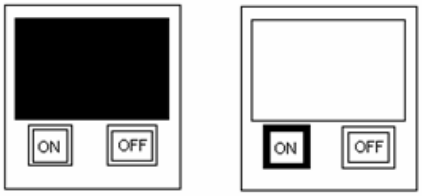
5.1 – State Machine
5.2 – Strategy
5.3 – Interpreter

Nokia Siemens Networks

5. Syntax and Input Analysis Design Patterns

5.1 – State Machine

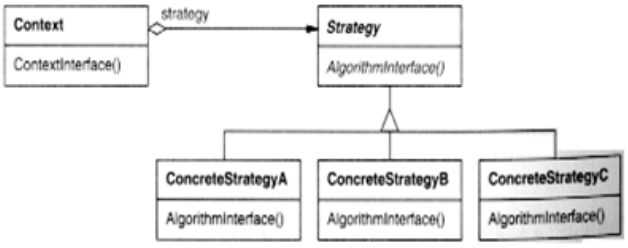
- Simplifies based-state operations
- Name:
  - State Machine
- Problem:
  - How to interact?
- Solution:
  - Allows viewing the software like an abstract "machine" that:
    - allows the input of a set of values
    - respond to the inputs it will perform one ore more actions
- Consequences:
  - Adaptability and efficiency
- Example (Lights):
  - input: On or Off; actions: display the light if On is selected



5. Syntax and Input Analysis Design Patterns (continued)

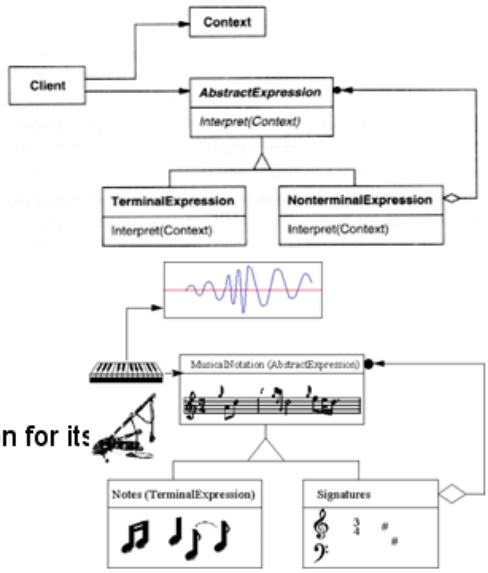
5.2 – Strategy

- Compact algorithms in subclasses of a common super class
- Name: Strategy
- Problem:
  - How to encapsulate algorithms in classes?
- Solution:
  - Strategy objects compact only one function (shown in the figure)
- Consequences:
  - Several small classes, leading to increased code size
- Many applications:
  - Vary an algorithm at runtime
  - A client can provide his own algorithm



5. Syntax and Input Analysis Design Patterns (continued)

5.3 – Interpreter



- This pattern allows interpretation of grammars
- Name:
  - Interpreter
- Problem:
  - How to interpret a grammar?
- Solution:
  - Given a language, define a representation for its
  - Define an interpreter
  - Sentences' interpretation using the representation defined earlier
- Consequences:
  - Useful for processing input streams – Efficiency is not a concern

Nokia Siemens Networks

Classes of Design Patterns (continued)

6. Roles Design Patterns:

- Dynamic changes to an object's apparent state or operations

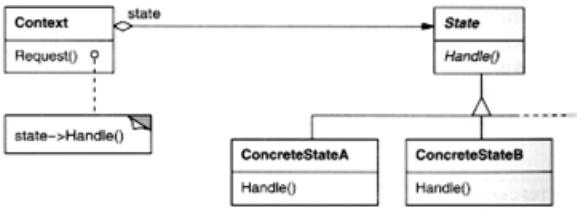
6.1 – State
6.2 – Decorator

Nokia Siemens Networks

6. Roles

6.1 – State

- When the object's internal state changes - Allows changing its behavior
- Name:
  - State
- Problem:
  - How the behavior of an object depends of the state?
- Solution:
  - Represent the different "states" of the state machine as derived classes of the State base class
- Consequences:
  - New roles can be added with no changes to the client
  - Aggregation allows the client assume several roles at once

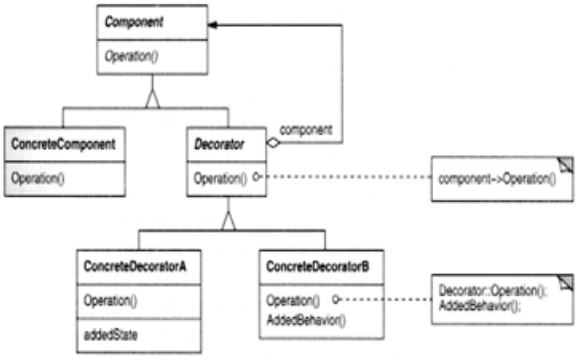


Nokia Siemens Networks

6. Roles (continued)

6.2 Decorator

- Add properties and operations to an object, in a transparent way, for its clients
- Name: Decorator
- Problem:
  - Functionality dynamically added to a stream
- Solution:
  - Subclass's instance delegates operations to the original object
- Consequences:
  - Easy to add functionality to a single object
  - Leave others like it unmodified



Nokia Siemens Networks

7. References

- FOWLER, Kim, "*Applying Object-Oriented Design Patterns: Hands-On*", Learning Tree International, 2006
- Ziv Research and Consulting, Classes and Objects Lecture 3, 1997, http://www.ics.uci.edu/~ziv/ooad/classes/, accessed September 18, 2006
- Index of /static/Offline/UML, Images, 1999, http://aliceinfo.cern.ch, accessed September 19, 2006
- Google, Pesquisa de imagens Google, 2006,http://images.google.pt, accessed September 19, 2006
- UML Notation, Atomic Object LLC, 2001, http://atomicobject.com, accessed September 18, 2006
- Directory Listing For /images, Images, 2005, http://www.javagen.com, accessed September 19, 2006
- Department of Computer Science, Johns Hopkins University, 2005, http://www.cs.jhu.edu, accessed September 18, 2006

Nokia Siemens Networks

7. References (continued)

- Mark Grand Java & OO Design Expert, Object Oriented Design & Analysis, 1998, http://www.mindspring.com/~mgrand, accessed September 19, 2006

Nokia Siemens Networks

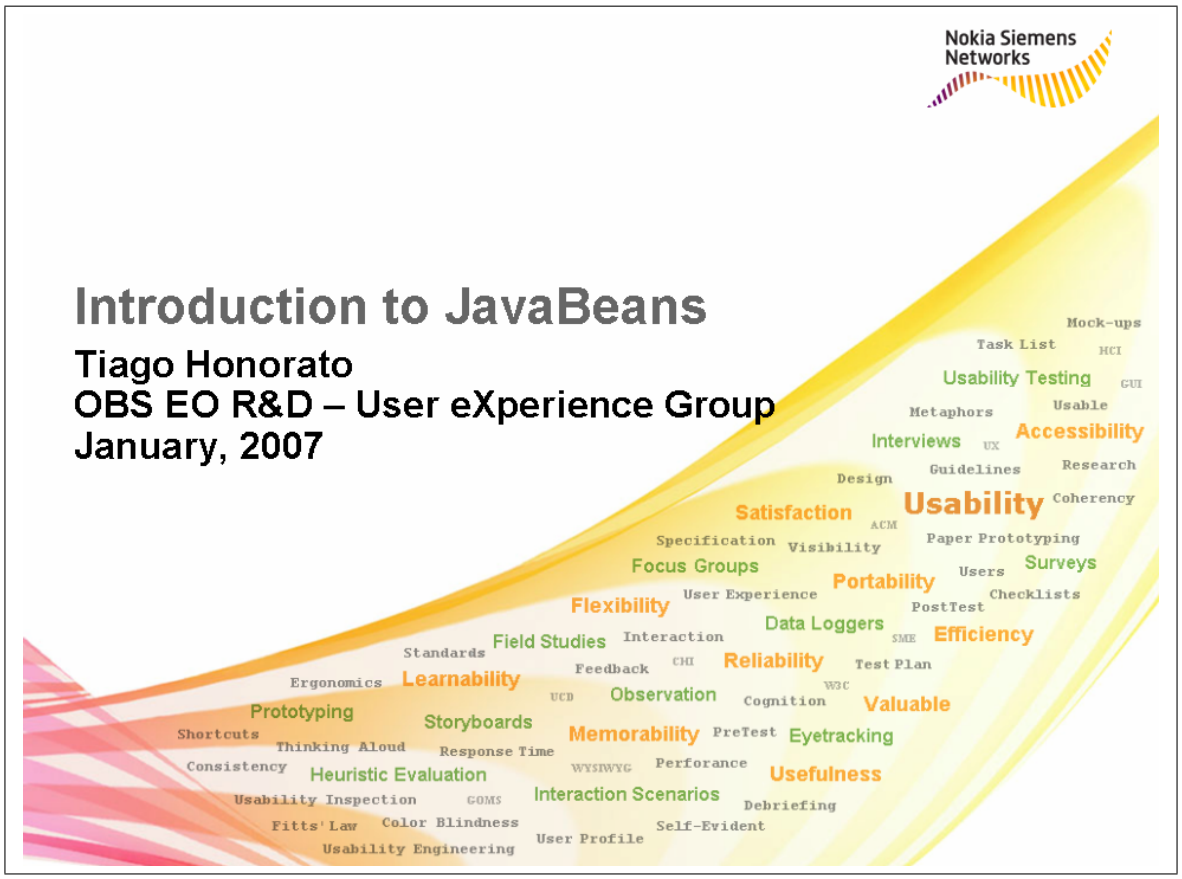
Contacts

Tiago Honorato
OBS EO R&D / Lisbon / Portugal
tiago.honorato@siemens.com

User eXperience Group
Nokia Siemens Networks Portugal, S.A.
Rua Irmãos Siemens, N.º1
2720-093 Alfragide
Fax: 351 – 21 416 7502

Apêndice C

Java Beans



Scope

These slides present an overview of J2EE JavaBeans

It contains the main contents to learn how Enterprise Java Beans (EJB) works

JavaBeans' architecture, components and examples are also shown here

Nokia Siemens Networks

Component-Based EJB Architecture

Nokia Siemens Networks

Architecture

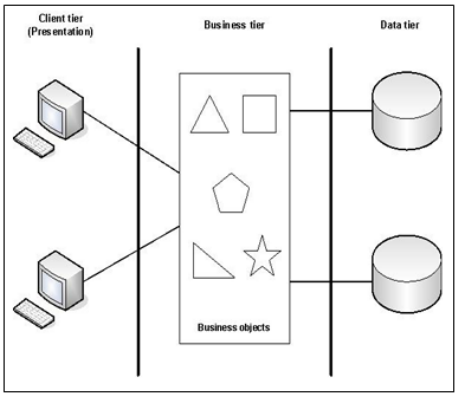
Components

Classifying EJBs

Nokia Siemens Networks

Three-Tier Architecture

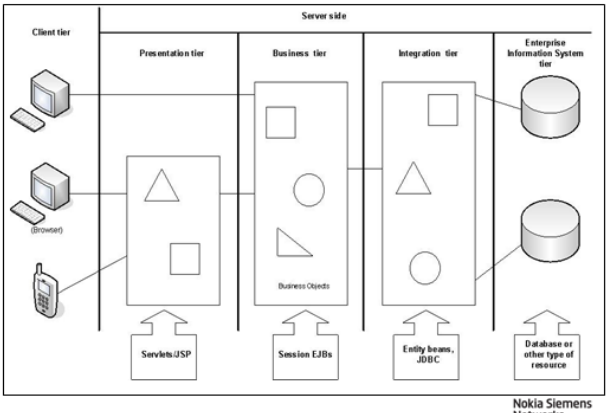
- Advantages:
  - Allows separation of presentation and business logic
  - Client is mostly responsible for presentation logic
  - Doesn't require client modification if middle tier is changed
- Disadvantages:
  - Lack of application portability
  - Integrate applications from different vendors



Nokia Siemens Networks

J2EE Architecture

- Presentation tier includes servlets, JSP, XML, XSL
- Business tier includes session EJBs (Business logic is implemented)
- Integration tier allows access and updating of various resources
- EIS tier represents that data



Nokia Siemens Networks

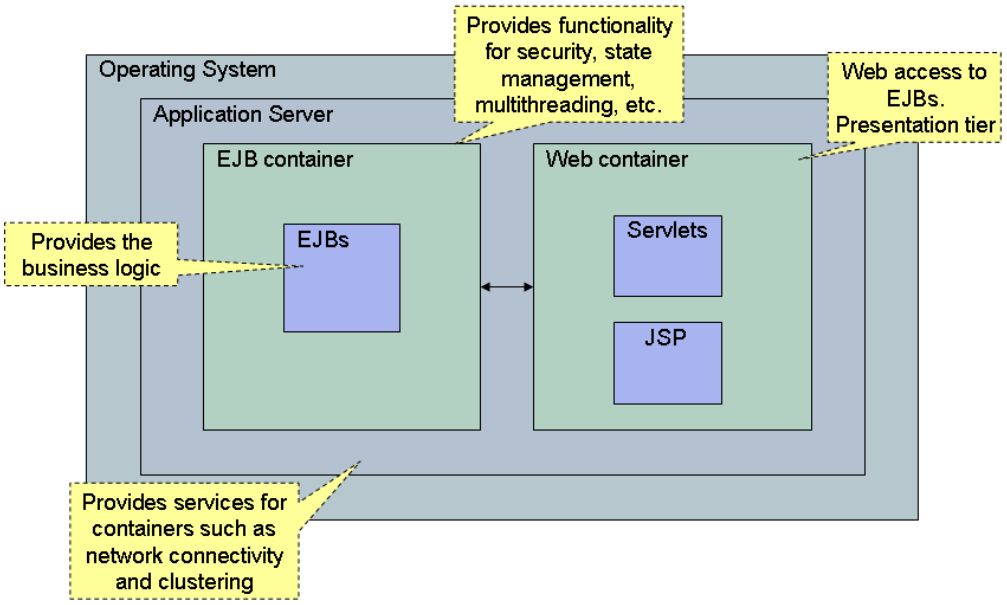
Architecture

Components

Classifying EJBs

Nokia Siemens Networks

General Structure



Application Server

- Provide component-based environment and support for:
  - Web container (Servlets, Java Server Pages (JSP))
  - EJB container (Support for session, entity & message-driven beans)
- Web container and EJB container can provide other functionalities:
  - Java Naming and Directory Inteface (JNDI)
  - Java Database Conectivity (JDBC)
  - Java Message Service (JMS)
  - Java Authentication and Authorization Service (JAAS)
  - Etc.
- Can support a large amount of users
- Developers can ignore knowledge of the application server
- Developers can concentrate in creating business logic
  - Transactions, load balancing, etc., are handled by server

Application Server (2)

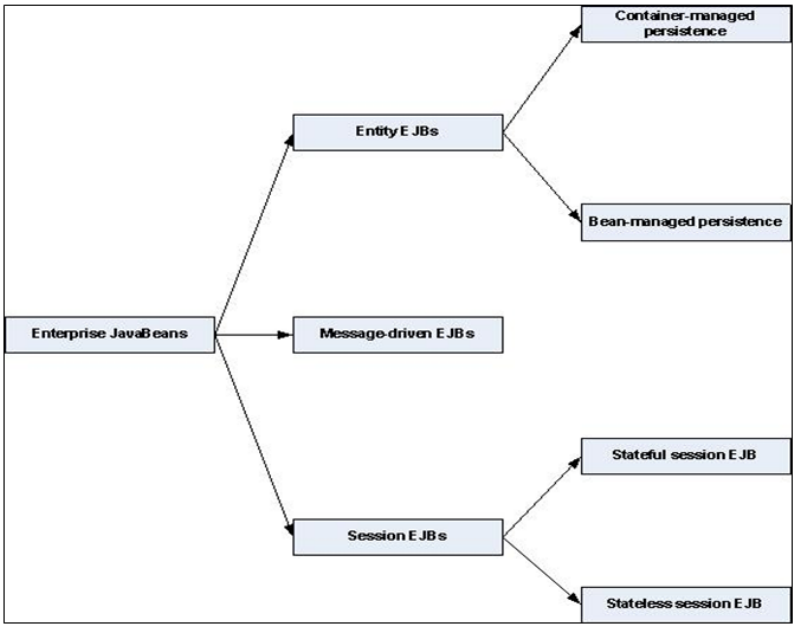
- Existing application servers
  - IBM WebSphere Application Server
    - High-performance application server
    - www.software.ibm.com
  - JBoss
    - Open source application server
    - Used in the Learning Tree course (course 576)
    - www.jboss.org
  - BEA Systems: BEA WebLogic Server
    - High-performance application server
    - Used in the Learning Tree course (course 576)
    - www.beasys.com

Architecture

Components

Classifying EJBs

EJB Classification



Nokia Siemens Networks

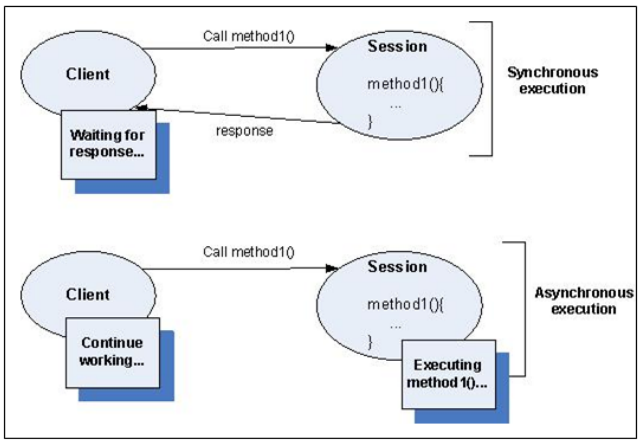
Entity EJB

- Entity EJB provides:
  - Object view of data in DB
  - Shared access to multiple users
  - Long live (since the data remains in DB)
- Bean-managed persistence (BMP):
  - Developer is responsible for writing the interface to DB
  - Easy integration with legacy systems
  - Can get very complex to maintain
- Container-managed persistence (CMP):
  - Container tools generate all code JDBC
  - Bean isn't tightly coupled with DB
  - Difficult dynamic SQL

Nokia Siemens Networks

Message-Driven EJB

- Characteristics
  - Provides component-model around Java Messaging Service
  - Asynchronous execution of business logic
  - Short-lived
  - Stateless
  - Can update data in DB
  - Can be transaction aware



Session EJB

- Characteristics
  - Implements business logic
  - Can access and update DB data
  - If container goes down, Session EJB "dies"
- Stateful session EJB
  - Maintains conversational state with a single client
  - Executes on behalf of the client
  - More complex then Stateless session EJB to develop
- Stateless session EJB
  - Easy to develop
  - Doesn't maintain conversational state
  - Very efficient

Client-EJBs interaction Example

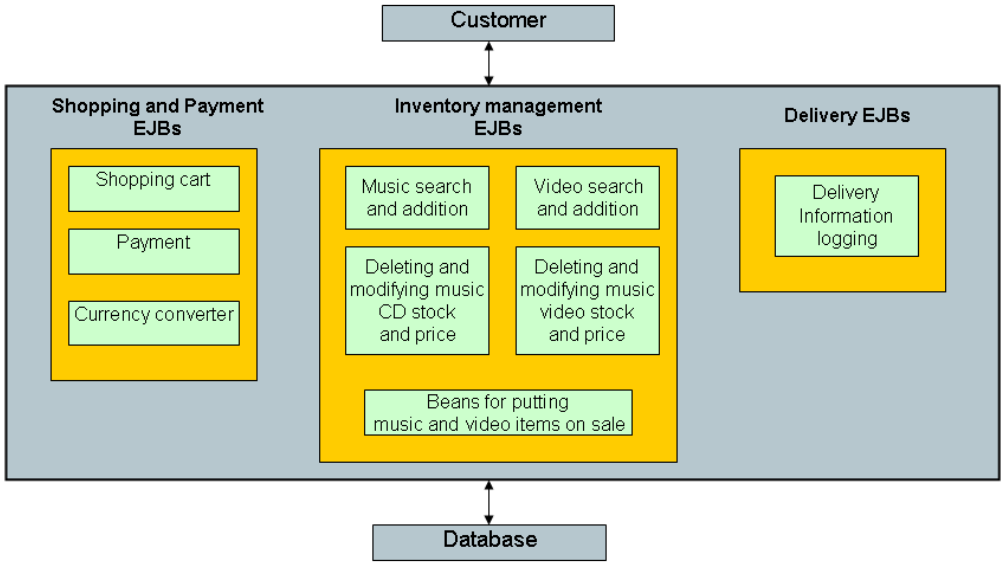
Nokia Siemens Networks

Video and music store

Client - EJBs

Nokia Siemens Networks

Video & Music Store Example - Using EJBs

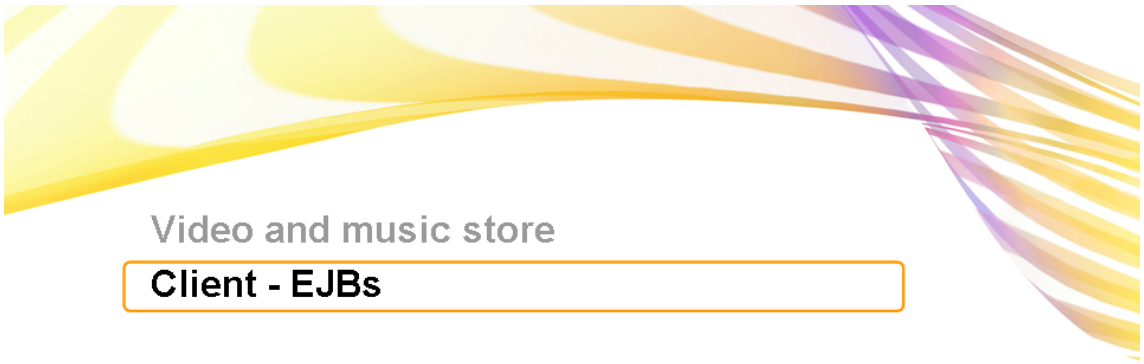


Video & Music Store Example - Using EJBs (2)

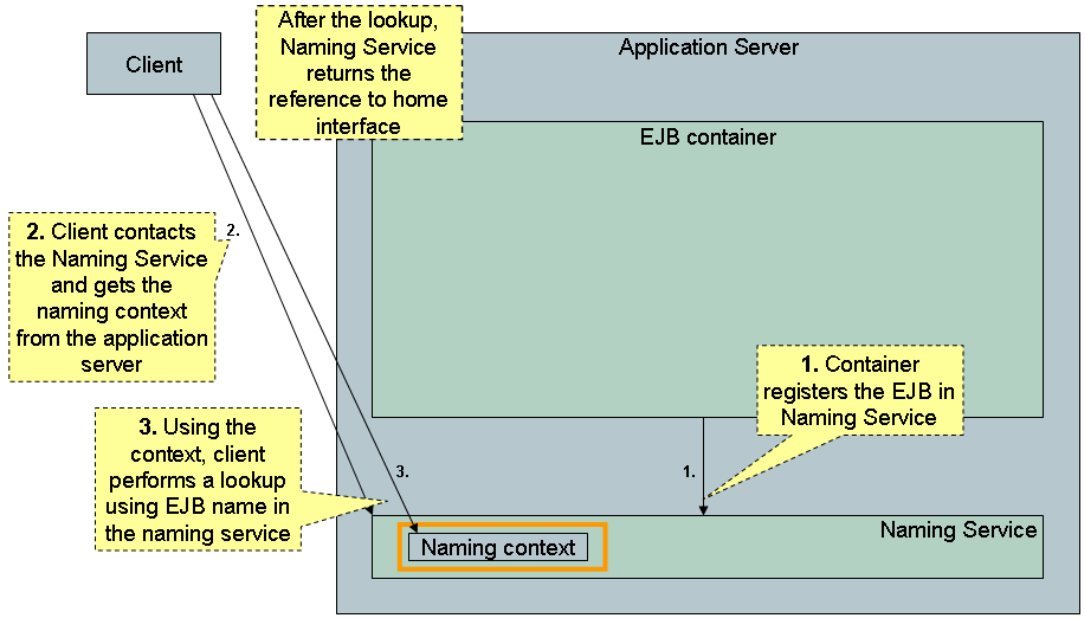
- The shopping cart should be a **Stateful session EJB**
  - It should remember the items bought by costumer during the session
  - One shopping cart should be associated with every costumer
- The payment should be a a **Stateless session EJB**
  - It allows the costumer to pay for the items
  - When a client does the payment, the bean doesn't have to remember anything about the customer

Video & Music Store Example - Using EJBs (3)

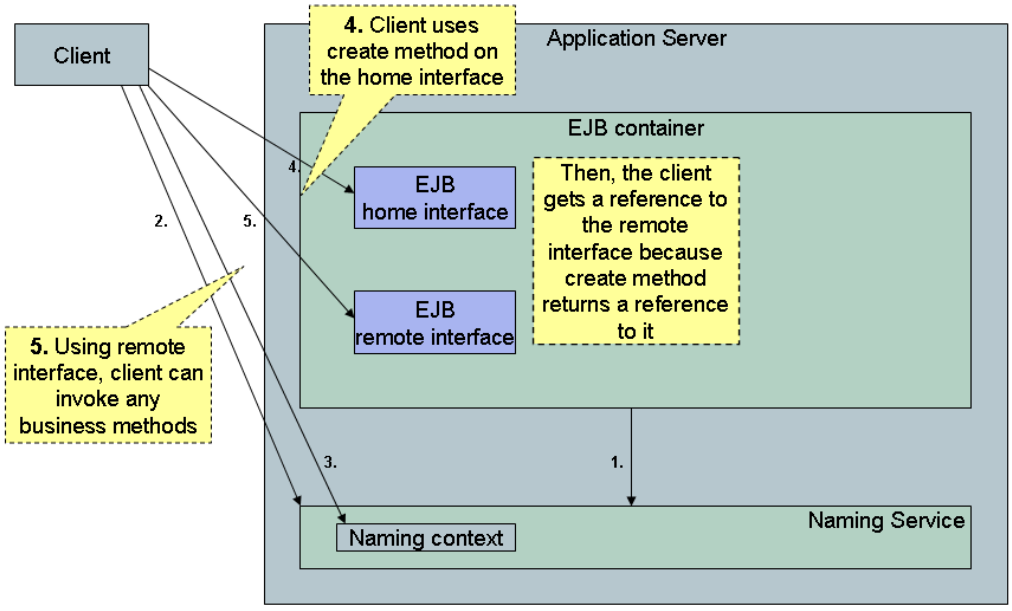
- The search and add music/video sould be represented by an **Entity bean (Container-Managed Persistence)**
  - It allows different types of searches on music/video in the DB
  - It can add new music CDs in the DB
- The prices' update can be made using **Message-Driven EJB**



Client and EJB interaction



Client and EJB interaction (2)

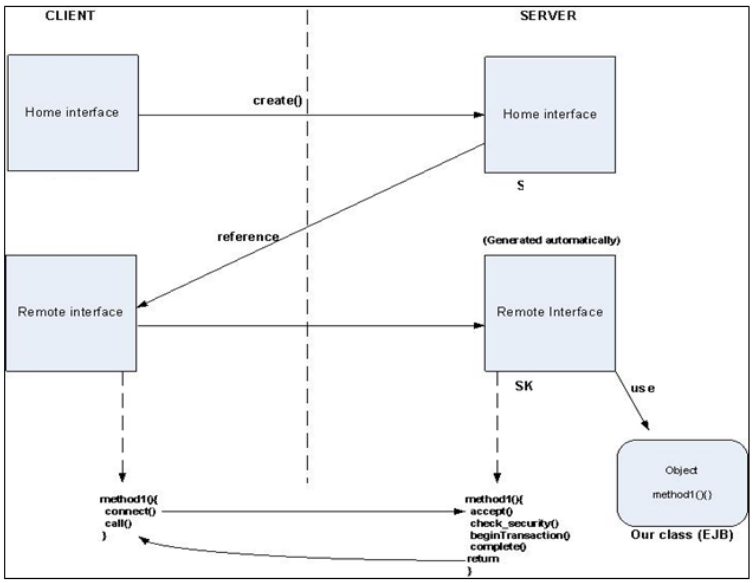


Client and EJB interaction (3)

- Home interface
  - Defines methods to create, remove or find an EJB
  - Container creates the class that implements home interface and generates a unique id for each EJB
- Remote interface
  - Defines business methods called by the client
  - The developer defines what method will be provided
  - Container creates a class that implements the remote interface

Nokia Siemens Networks

Client and EJB interaction (4)



Nokia Siemens Networks

Writing an EJB

Nokia Siemens Networks

Writing a Welcome Session EJB

Related technologies for building EJBs

Using JNDI

Using JDBC

Using DataSource

Nokia Siemens Networks

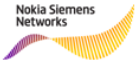
Writing a Welcome Session EJB

- Step 1: Write home interface
- Step 2: Write remote interface
- Step 3: Write bean's implementation
- Step 4: Compile home and remote interfaces and bean implementation class
- Step 5: Deploy the EJB
- Step 6: Write the client
- Step 7: Run the client

Writing a Welcome Session EJB (Step 1)

- Home interface
  - Allows the creation of a bean
  - Must have one create method and must be public
  - create method returns a remote interface
  - Example:

```
import java.rmi.RemoteException;
import javax.ejb.CreateException;
import javax.ejb.EJBHome;

public interface WelcomeHome extends EJBHome{

        public Welcome create() throws RemoteException, CreateException;

}
```

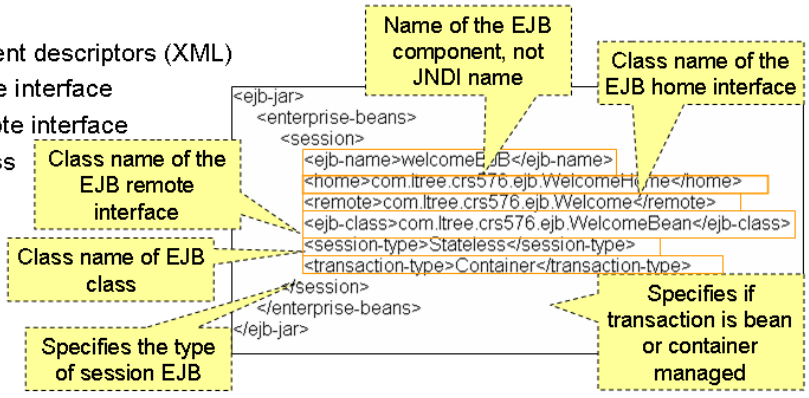
Writing a Welcome Session EJB (Step 2)

- Remote interface
  - All the methods that are supported in the EJB must be declared
  - All methods must throw RemoteException
  - Example:

```
import java.rmi.RemoteException;
import javax.ejb.EJBObject

public interface Welcome extends EJBObject{

        public Welcome getWelcome(String message) throws RemoteException;

}
```

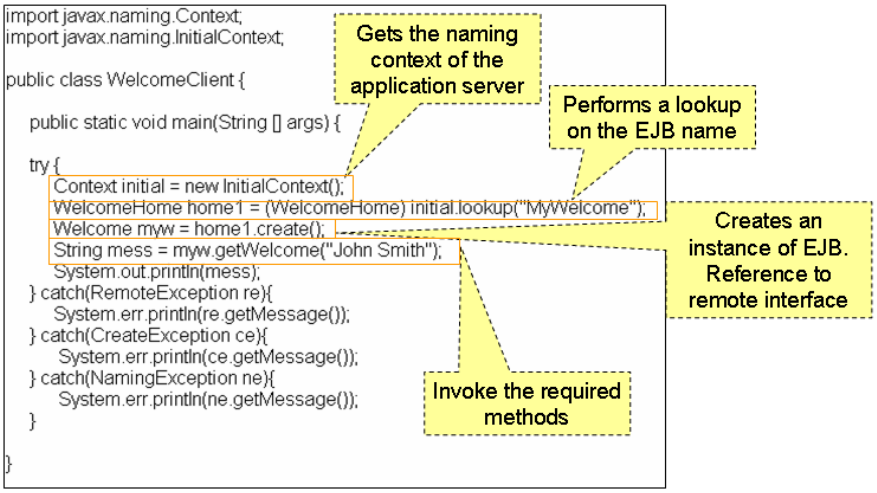

Writing a Welcome Session EJB (Step 4 and 5)

- Compiling:
  - javac Welcome.java WelcomeHome.java WelcomeBean.java
- Deployment
  - Process of installing an EJB in an EJB container
  - Requires:
    - Deployment descriptors (XML)
    - EJB home interface
    - EJB remote interface
    - Bean class
    - Example:



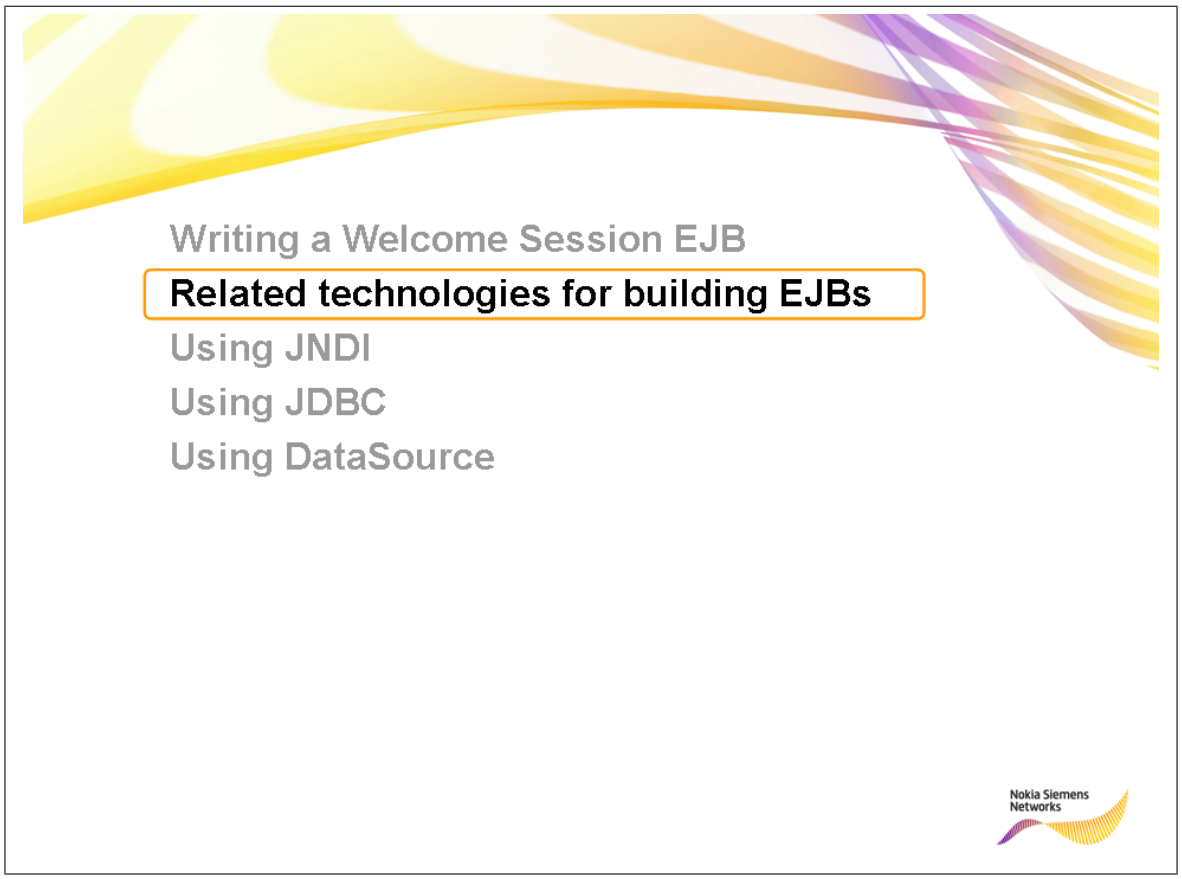
Writing a Welcome Session EJB (Step 6)

- Client example:



Writing a Welcome Session EJB (Step 3)

- Running the client
  - Command line> java WelcomeClient
- The output
  - "Welcome John Smith"



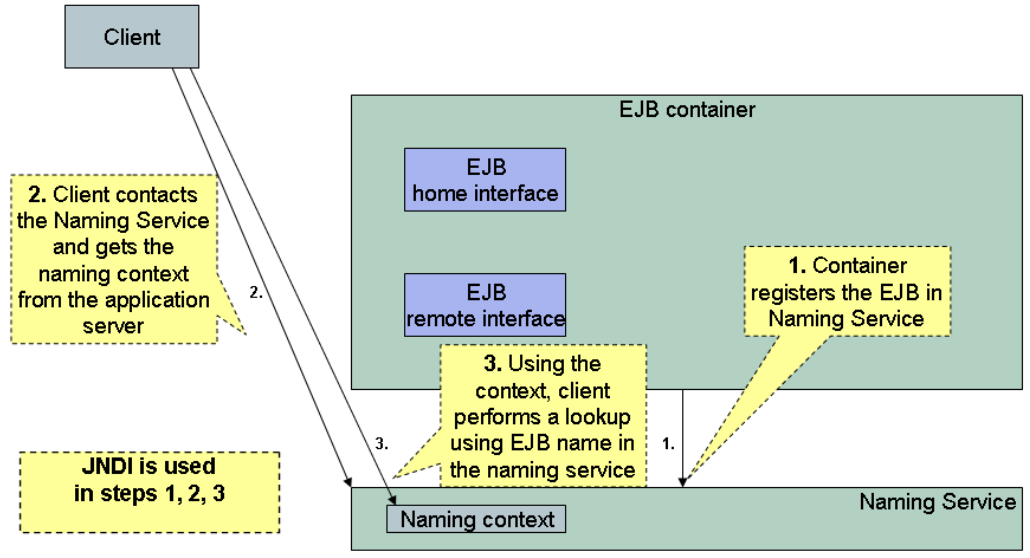
Related technologies for building EJBs

- Java Naming and Directory Interface (JNDI)
  - Naming Services
  - Directory Services
  - Associates a name with a service
  - Allows registering a service by given a name
- Java Database Connectivity (JDBC)
  - JDBC drivers are needed to communicate to the database
  - Can create statements and execute SQL queries
  - Can use DataSource to increase performance

Writing a Welcome Session EJB
Related technologies for building EJBs
Using JNDI
Using JDBC
Using DataSource

Nokia Siemens Networks

Using JNDI



Nokia Siemens Networks

Related technologies for building EJBs

- Step 0: Initializing parameteres for naming service access
  - Performed on the client and the service provider side
  - Parameters are passed using Hashtable class
  - Requires three activities (A)
    - Select a service provider for registration and access (A1)

```
Hashtable ht = new Hashtable();
ht.put(Context.INITIAL_CONTEXT_FACTORY, "weblogic.jndi.WLInitialContextFactory");
```

    - Specify initial values for configuration (A2)

```
ht.put(Context.PROVIDER_URL, "t3://localhost:7001");
ht.put(Context.SECURITY_PRINCIPAL, "userid");
ht.put(Context.SECURITY_CREDENTIALS, "password");
```

    - Invoke InitialContext constructor (A3)

```
Context ctx = new InitialContext(ht);
```


Related technologies for building EJBs

- Step 1: Registering a service
  - "sname" as the user-specified name
  - sObj as service object
  - Two ways to register
    - Any previous binding with sname will be removed
    - Overwrite an existing bind

```
ctx.rebind("sname", sObj);
```

    - Similar to rebind(...) but throws a NamedAlreadyBoundException

```
ctx.bind("sname", sObj);
```

- Step 2: Service lookup and invocation of services
  - lookup can be performed as:

```
Object obj = ctx.lookup("sname");
```

Writing a Welcome Session EJB
Related technologies for building EJBs
Using JNDI
Using JDBC
Using DataSource

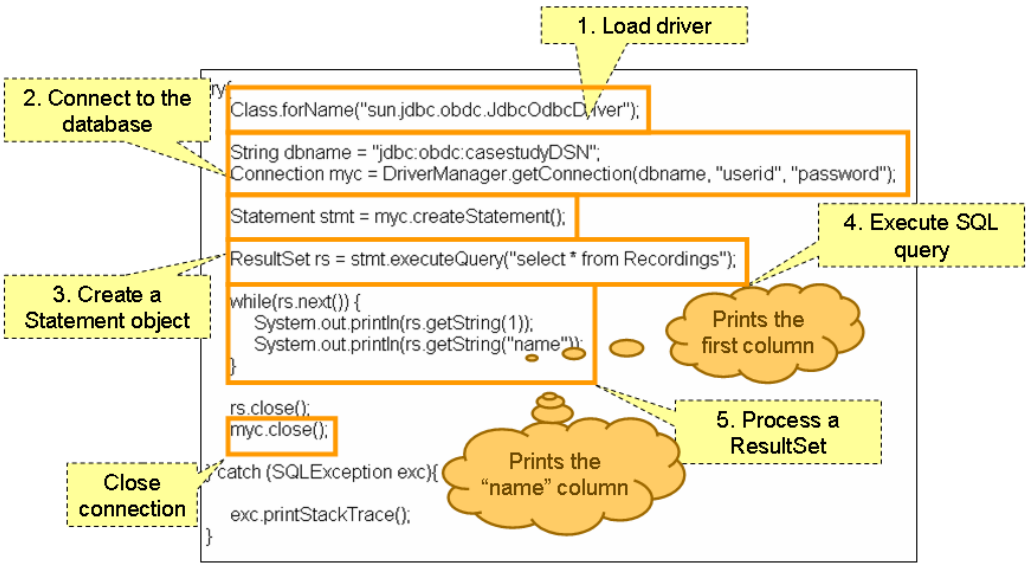
Nokia Siemens Networks

Using JDBC

- Main steps
  - Load a driver
  - Connect to the database
  - Create a Statement object
  - Execute a SQL query
  - Process a ResultSet

Nokia Siemens Networks

Using JDBC (2)



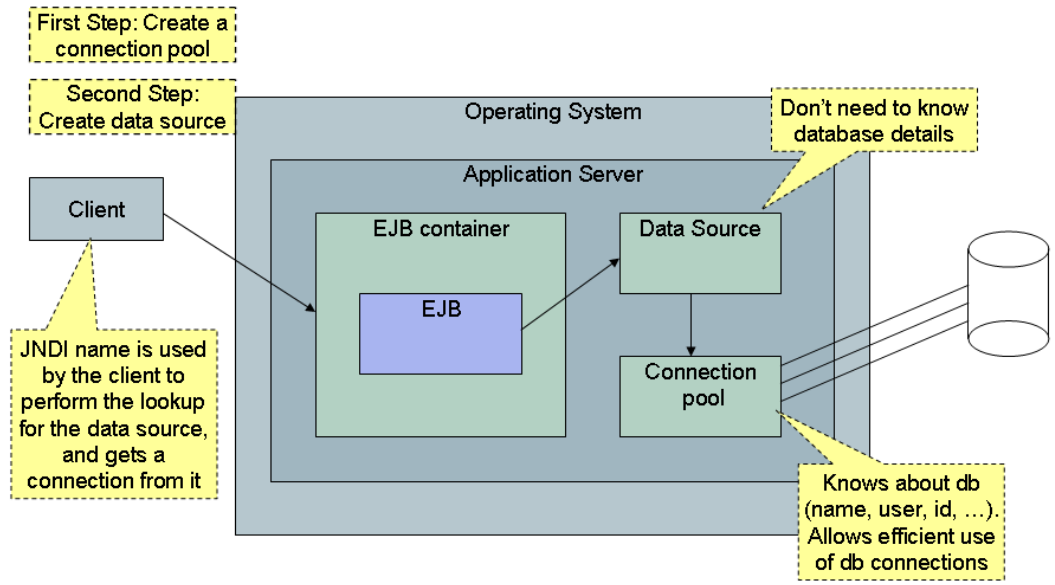
For internal use
42 © Nokia Siemens Networks Introduction to JavaBeans / Tiago Honorato / January 2007

Nokia Siemens Networks

Writing a Welcome Session EJB
Related technologies for building EJBs
Using JNDI
Using JDBC
Using DataSource

Nokia Siemens Networks

Using DataSource



Using DataSource (2)

- Data Source - Code Sample

```
private java.sql.Connection getConnection() throws SQLException {

    InitialContext initCtx = null;
    DataSource ds = null;
    String url = "t3://localhost:7001"
    Connection dBConn = null;

    try{
        Hashtable h = new Hashtable();
        h.put(Context.INITIAL_CONTEXT_FACTORY, "weblogic.jndi.WLInitialContextFactory");
        h.put(Context.PROVIDER_URL, url);
        initCtx = new InitialContext (h);
        ds = (javax.sql.DataSource) initCtx.lookup("MusicDataSource");
        dBConn = ds.getConnection();

    } catch(Exception ne){
        System.out.println("Unable to get a connection");

    }

return dBConn;
}
```

Using JNDI services

Get the connection

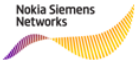
Performing the lookup

Conclusion

- Enterprise Java Beans (EJB) enables software developers to design and create reusable pieces of software
- EJB's Properties
  - Distributed, Reliable, Secure, Scalable, Persistence, Transactional, Concurrent
- To spend less time building these facilities:
  - Delegate the above activities to Java's server-side component infrastructure (EJB + EJB container + Application server)
- If two of these properties are needed, the use of JavaBeans is recommended

References

- Czarnecki, Chris, "*Enterprise JavaBeans 2.1: Hands-On*", Learning Tree International, 2006
- Google, Pesquisa Google, 2006, http://www.google.pt, accessed January 14, 2007



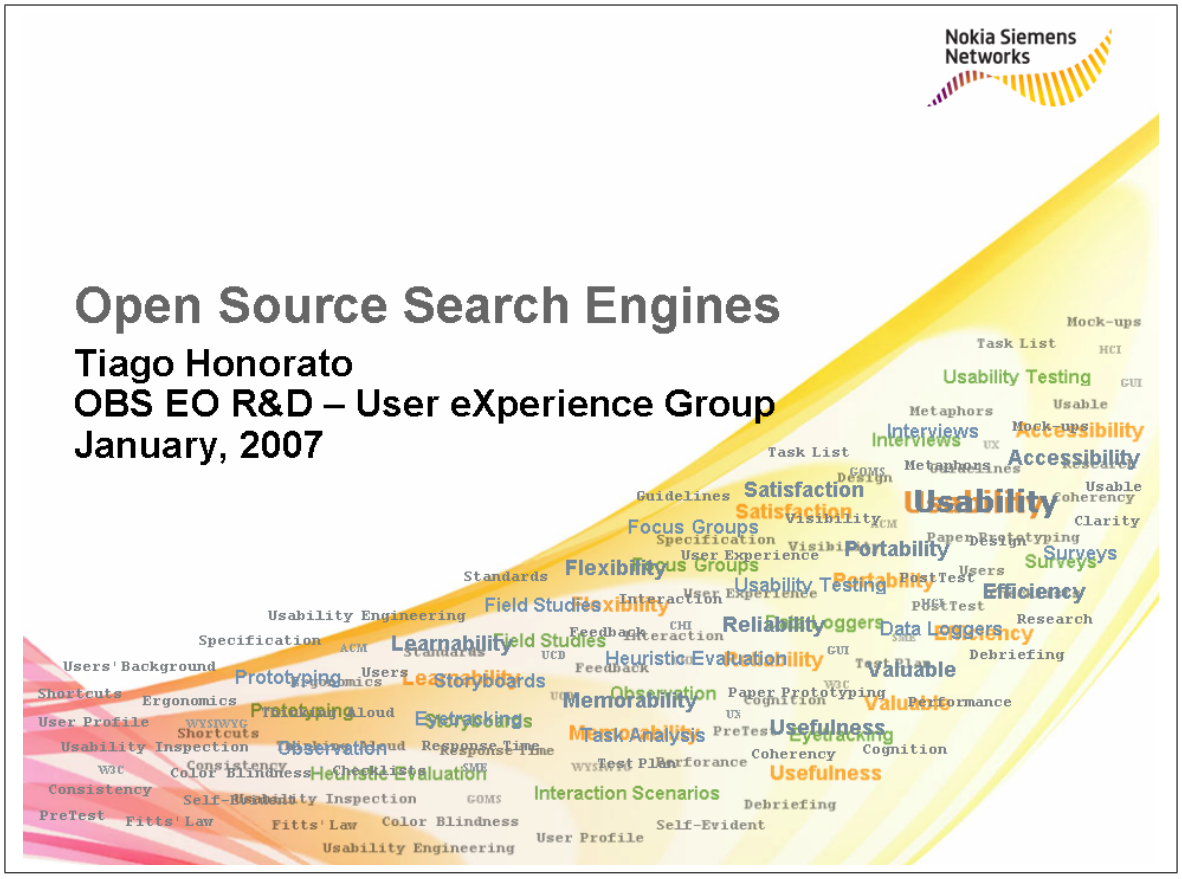
Contacts

Tiago Honorato
OBS EO R&D / Lisbon / Portugal
tiago.honorato@siemens.com

User eXperience Group
Nokia Siemens Networks Portugal, S.A.
Rua Irmãos Siemens, N.º1
2720-093 Alfragide
Fax: 351 – 21 416 7502

Apêndice D

Search Engines



Introduction / Contents

• These slides present an analysis and overview of three open source search engines.

• User interface examples are shown and a comparison is provided together with example applications.

Collecting search engines

- A research of the existing search engines was the first thing to do [1]
- Then, only the open source search engines were considered (total: 37)
- Some filters were applied (programming language, operating system, future development, etc.)
- Resulting three search engines applications
  - DocSearcher
  - Regain
  - Zilverline
- All of these search engines are written in Java and based on Lucene

Lucene

- Lucene is an open source search engine library
- Written in Java
- Released under Apache Software License
- How it works?
  - There is a notion of document that contains fields of text
  - Therefore, text from PDF's, HTML, Microsoft word documents, etc., can be indexed because their textual information can be extracted

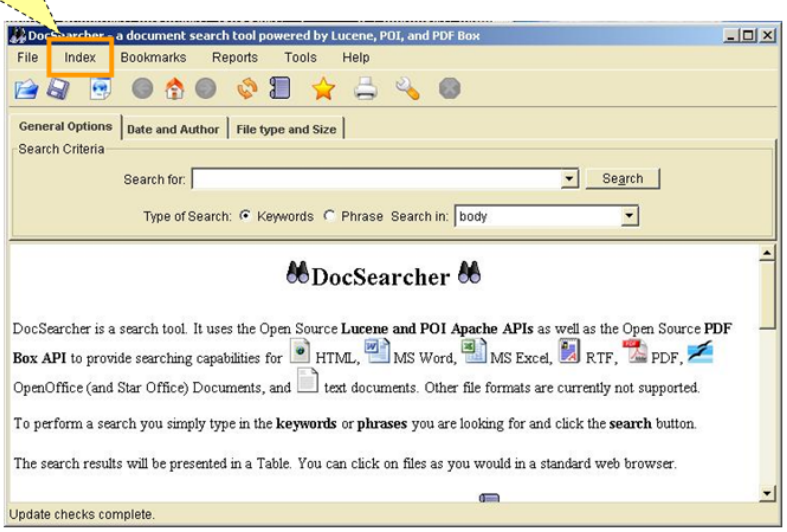
DocSearcher

- DocSearcher [2] is a search tool that uses Lucene, POI Apache [3] and PDF Box APIs [4]
- Desktop application that is portable to any platform
- Advantages
  - Searches in HTML, TXT, PDF, RTF, Word, Excel, Open Office files
  - Runs in all systems that have Java technology installed
  - GNU General Public License (GPL)
  - Can create several indexes
- Disadvantages
  - No highlighting
  - Other file formats are currently not supported (ZIP, RAR, Powerpoint)

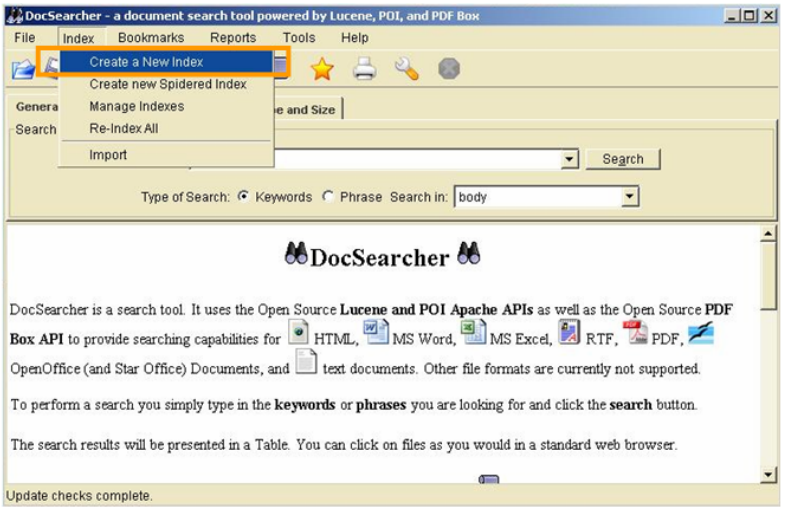
DocSearcher

Let's create an index

DocSearcher Main Window

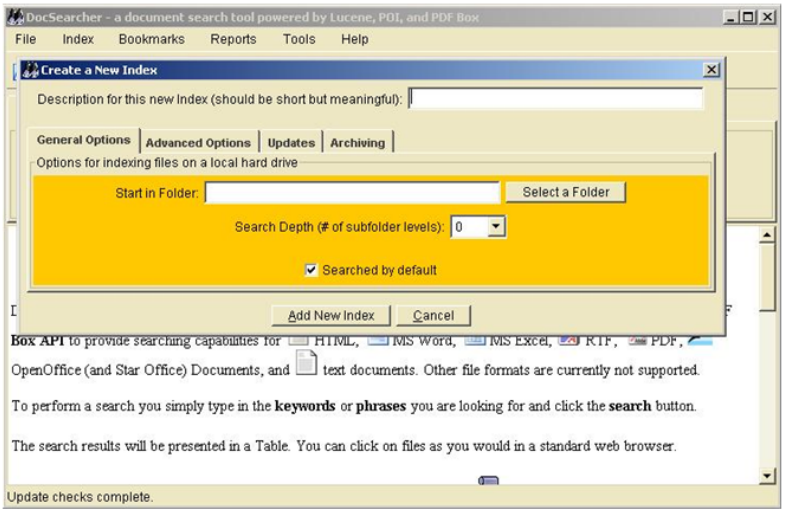


DocSearcher (2)



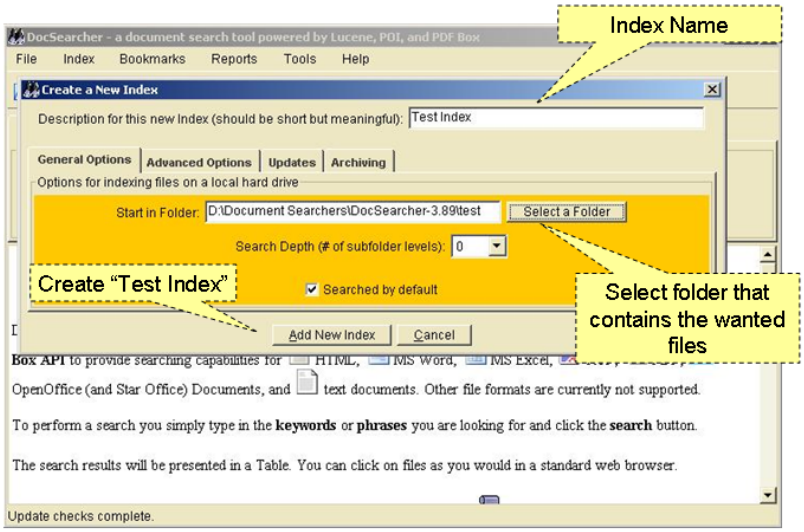
DocSearcher (3)

Index → Create new Index



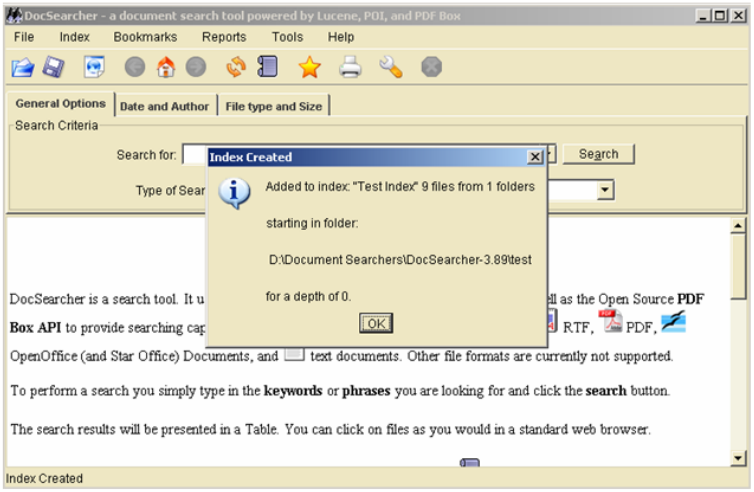
Nokia Siemens Networks

DocSearcher (4)

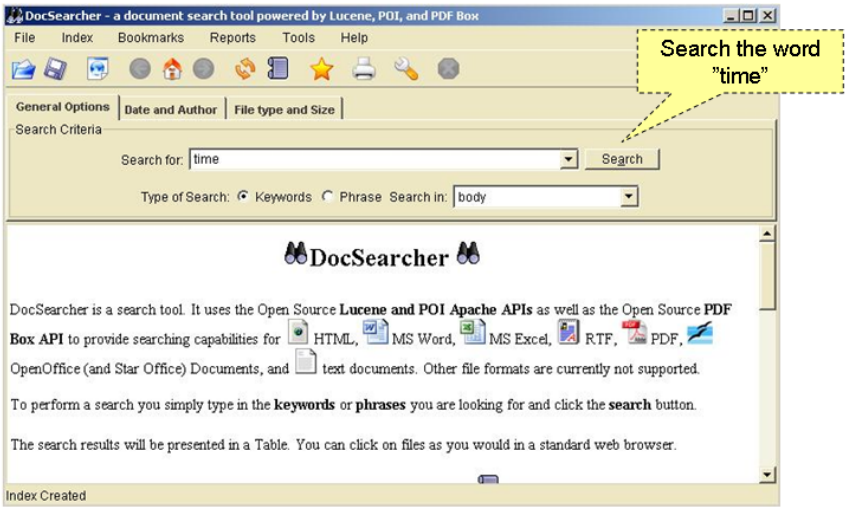


Nokia Siemens Networks

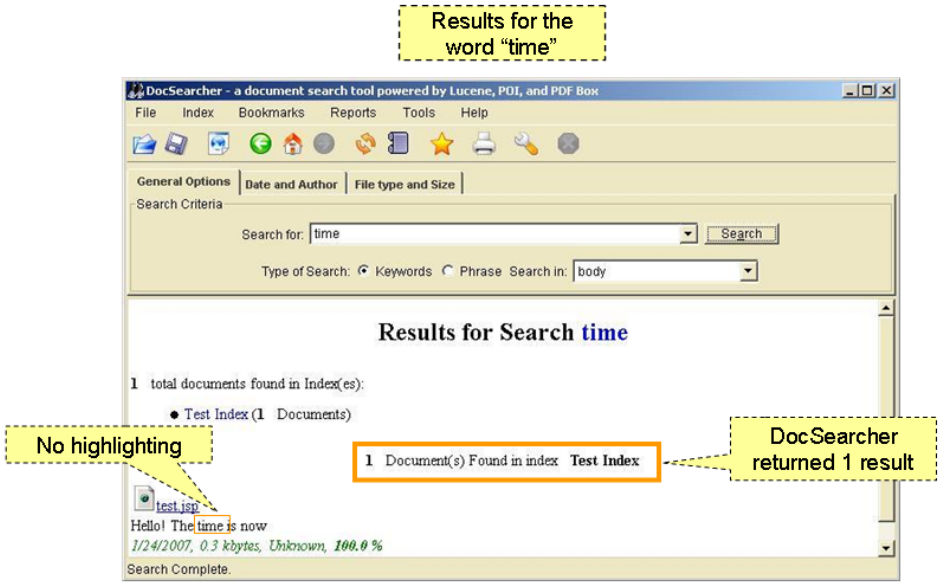
DocSearcher (5)



DocSearcher (6)



DocSearcher (7)

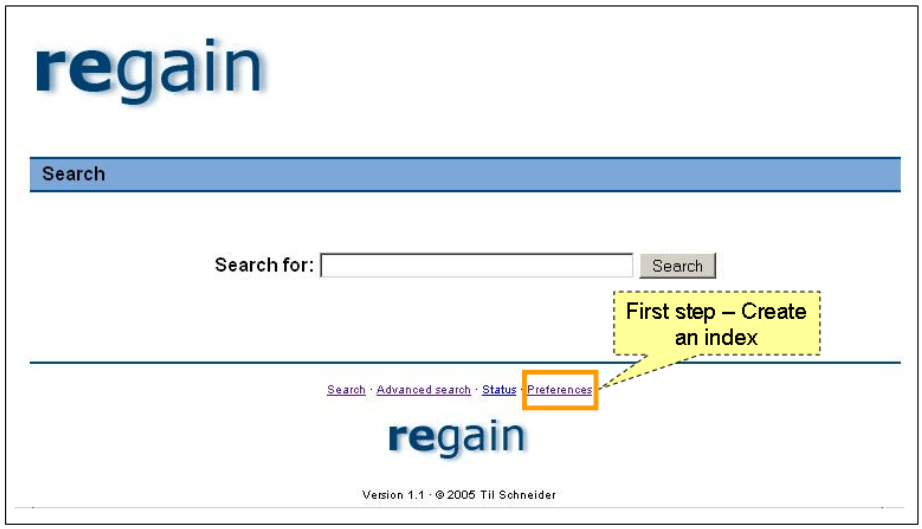


Regain

- Regain [5] is a search engine similar to web search engines like Google
- Advantages
  - Uses Java Server Pages (JSP)
  - Released under the open source license LGPL (Lesser General Public License)
  - Can search in HTML, TXT, PDF, RTF, Word, Excel, Powerpoint and JSP files
  - Indexing files faster than DocSearcher and Zilverline
  - Creation of several indexes is possible
- Disadvantages
  - No highlighting
  - Difficult development due to source code complexity
  - RAR and ZIP files not supported

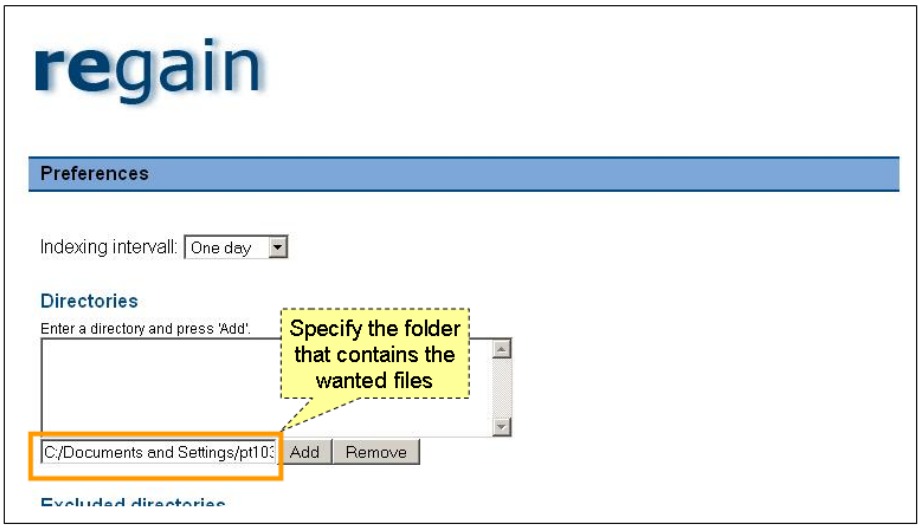
Regain

- Regain main window

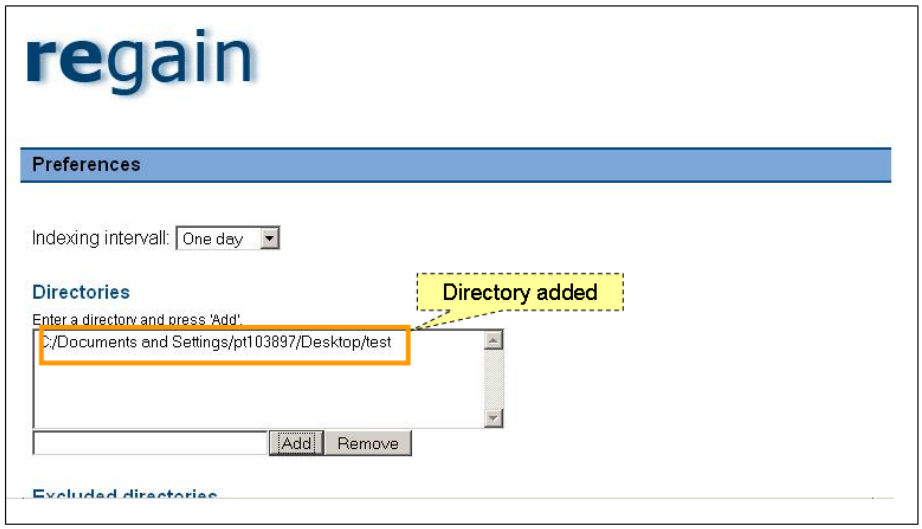


Regain (2)

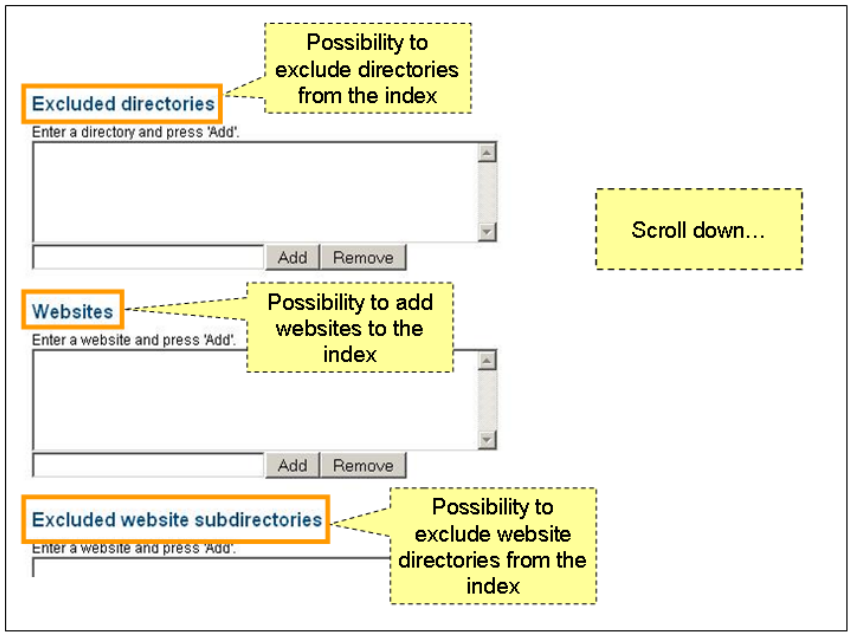
- Index



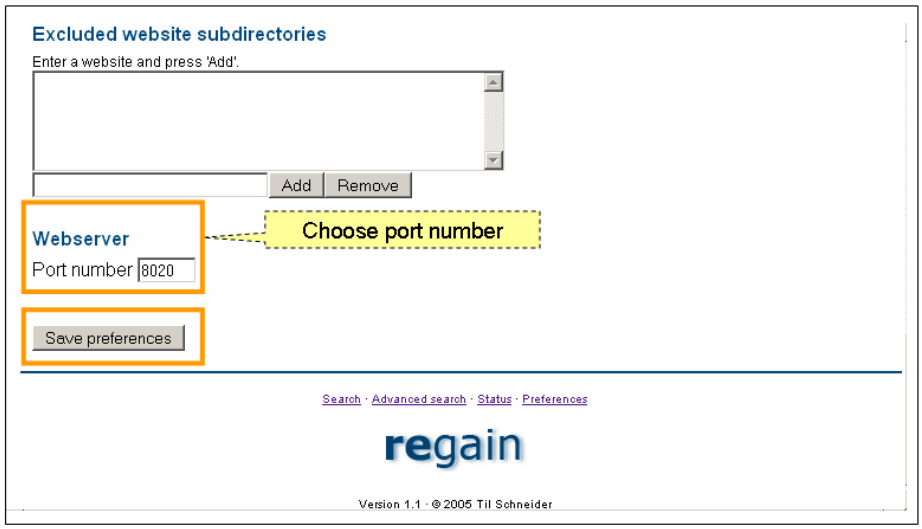
Regain (3)



Regain (4)

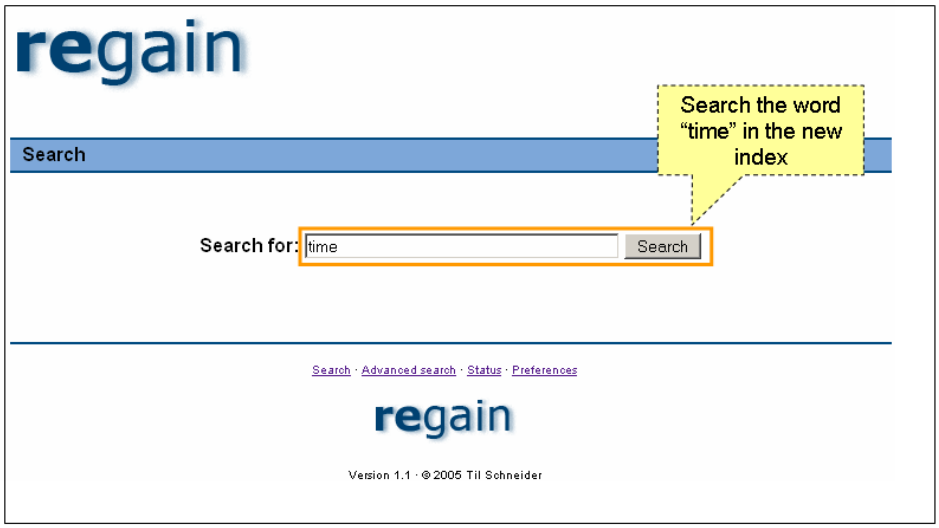


Regain (5)



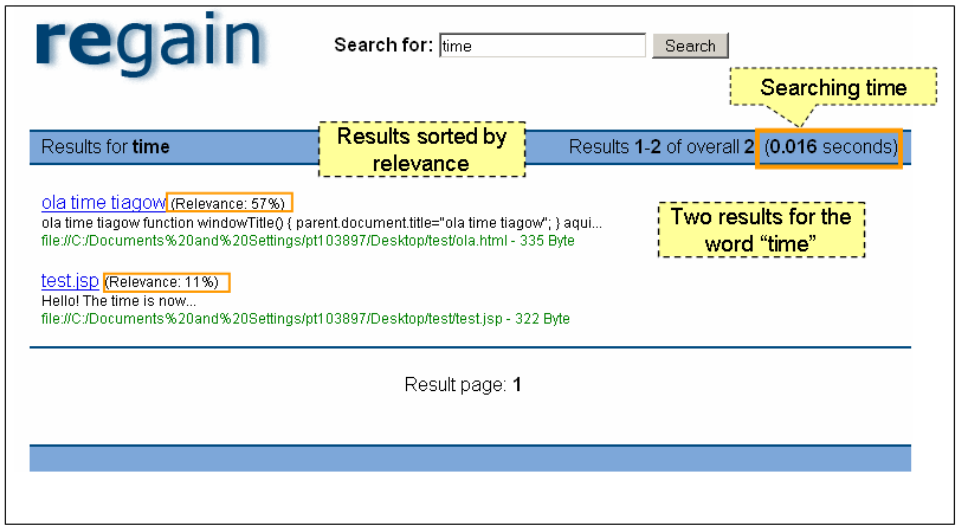
Nokia Siemens Networks

Regain (6)

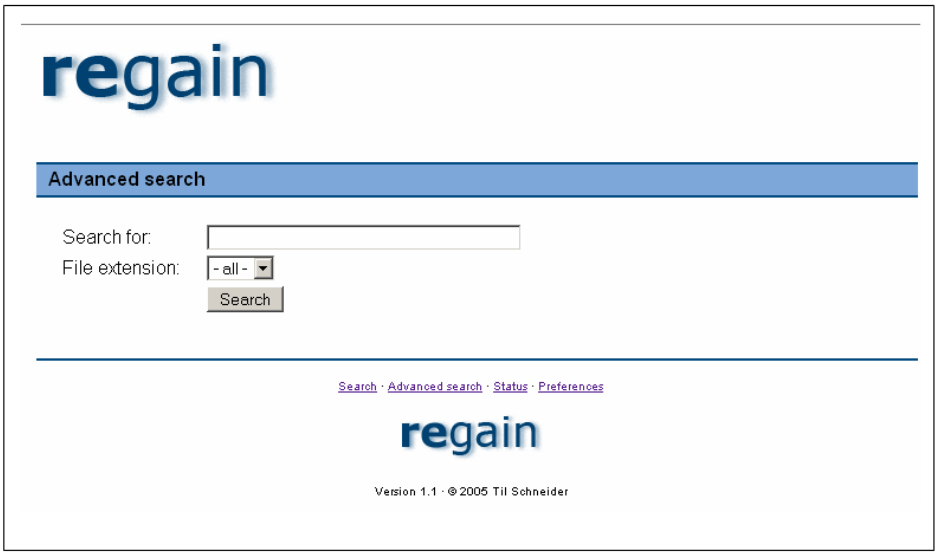


Nokia Siemens Networks

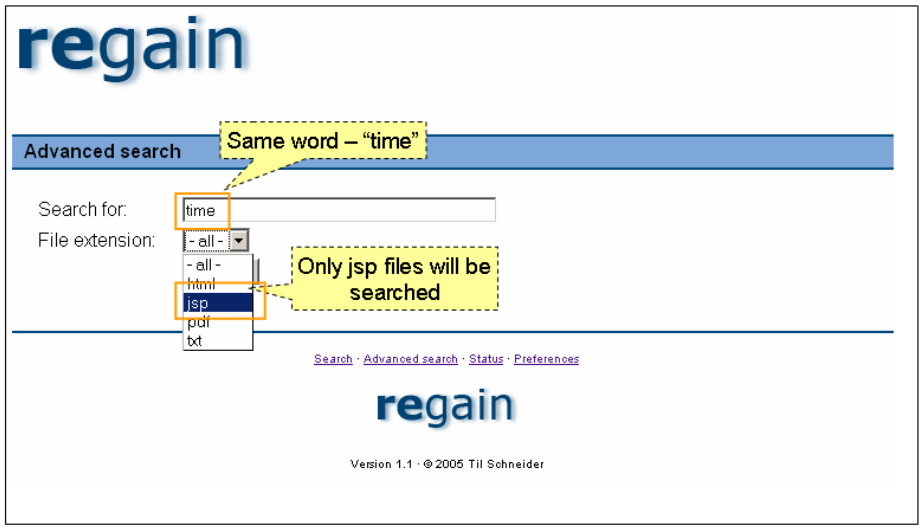
Regain (7)



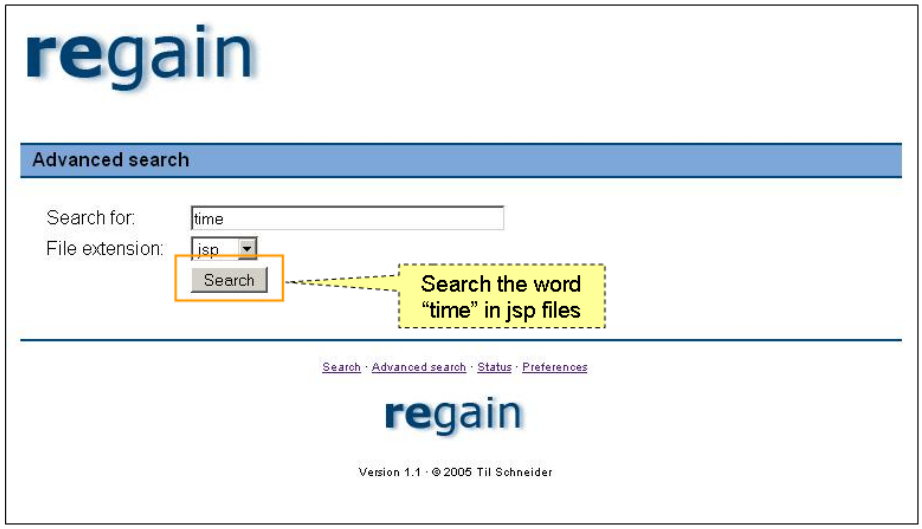
Regain



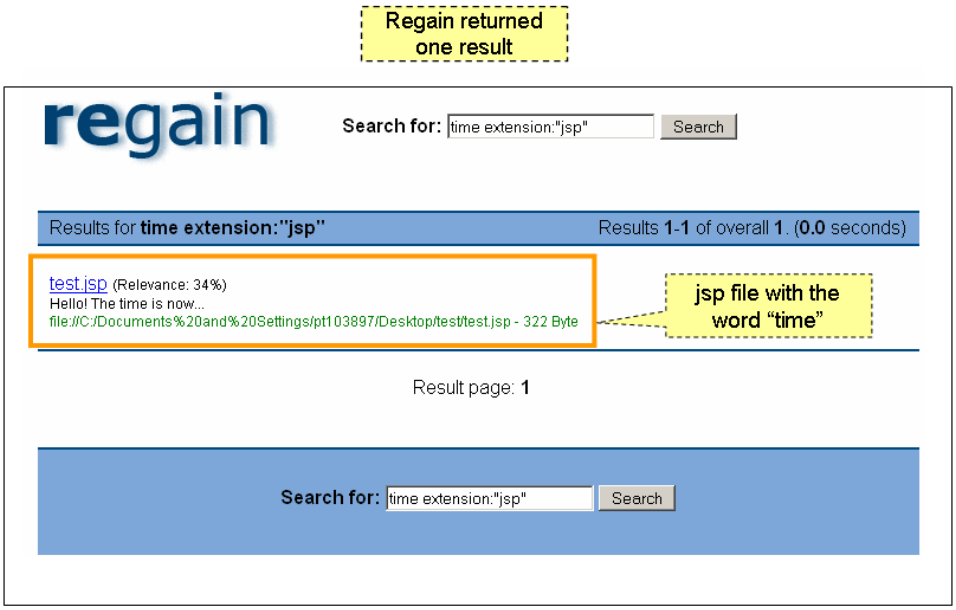
Regain (8)



Regain (9)



Regain (10)

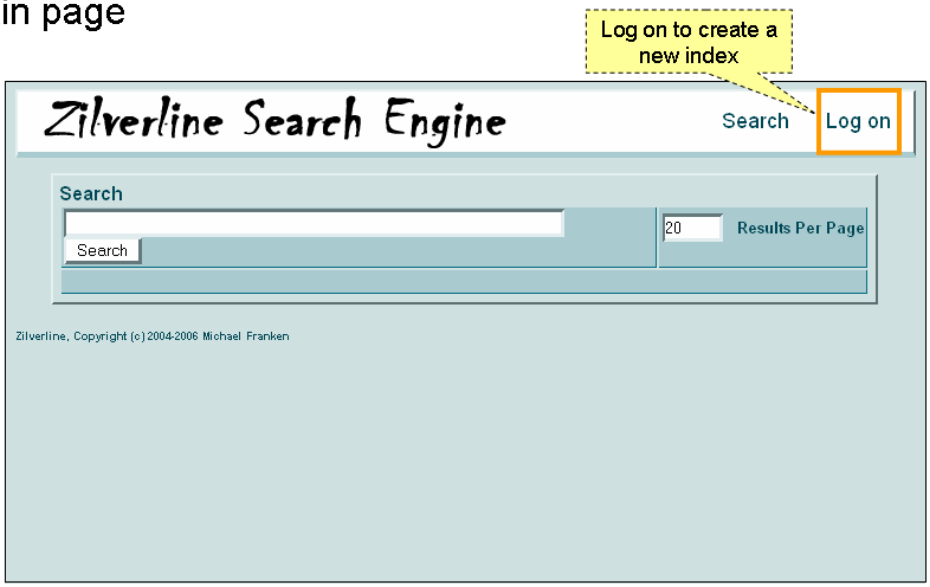


Zilverline

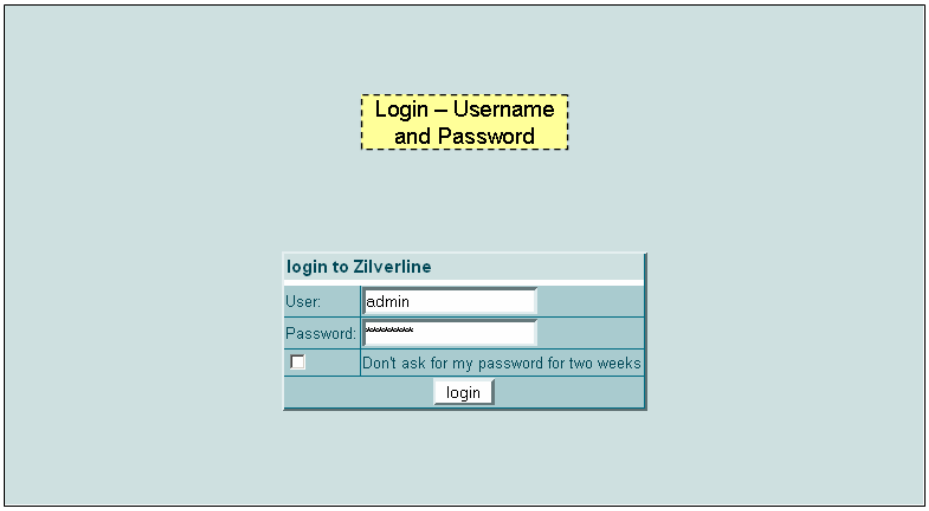
- Zilverline [6] is a search engine that offers web access to personal or intranet content
- Advantages
  - Zilverline extracts content from PDF, Word, Excel, Powerpoint, RTF, TXT, JAVA, CHM as well as ZIP, RAR, and many other archives
  - Can index and search several directories
  - Highlights the searched words in the result
- Disadvantages
  - Indexing time
  - Not free for commercial use

Zilverline (2)

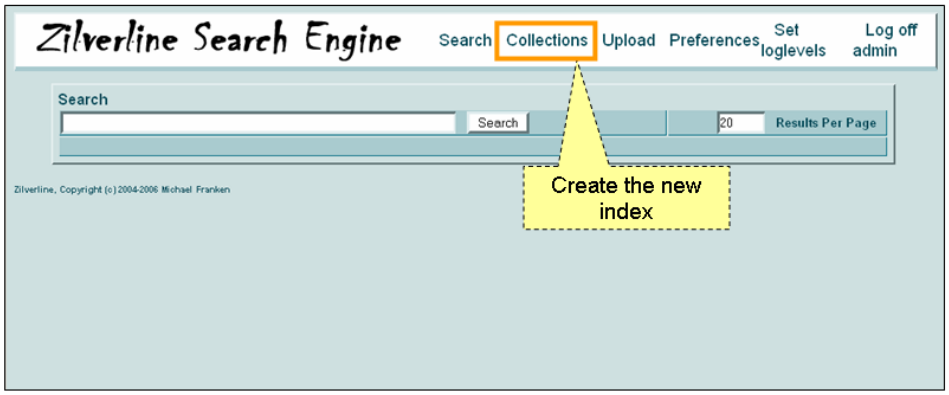
- Main page



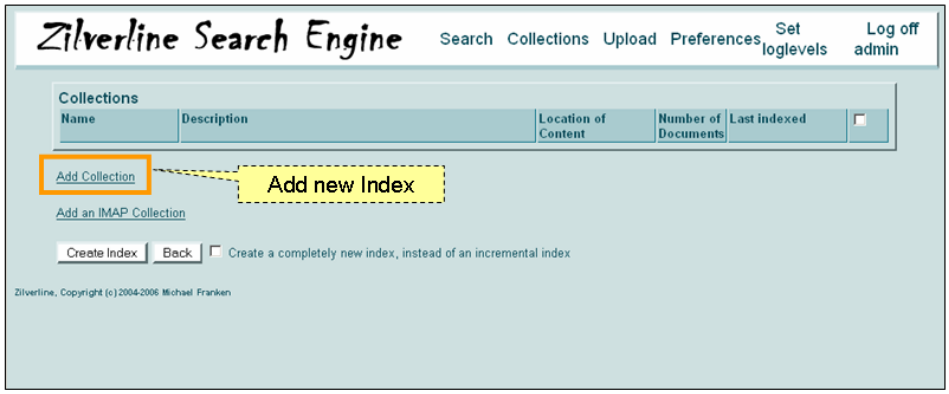
Zilverline (3)



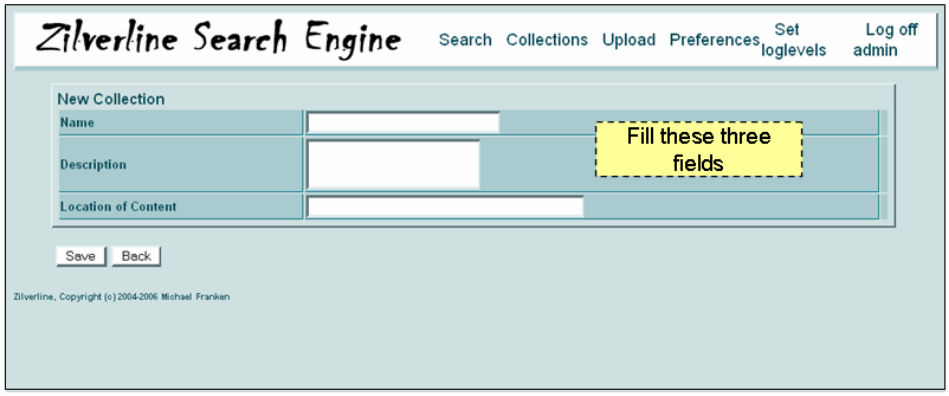
Zilverline (3)



Zilverline (4)

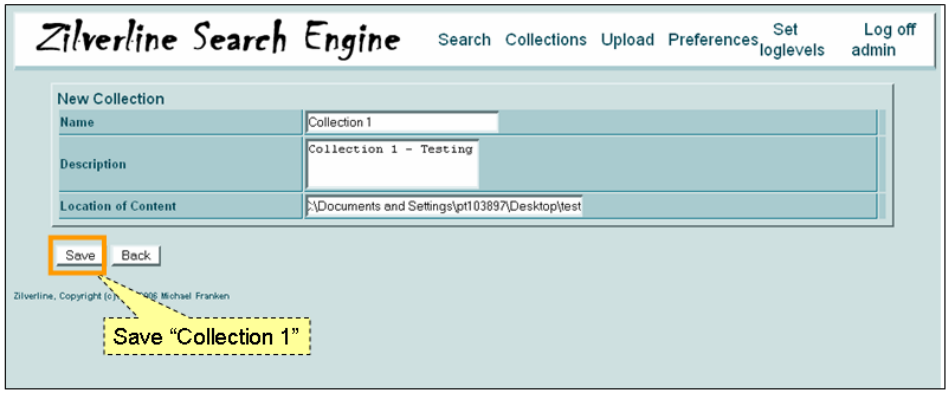


Zilverline (5)



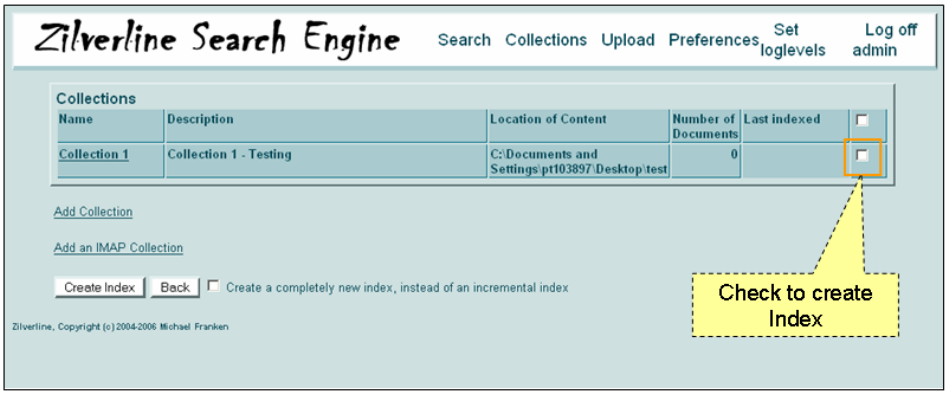
Nokia Siemens Networks

Zilverline (6)



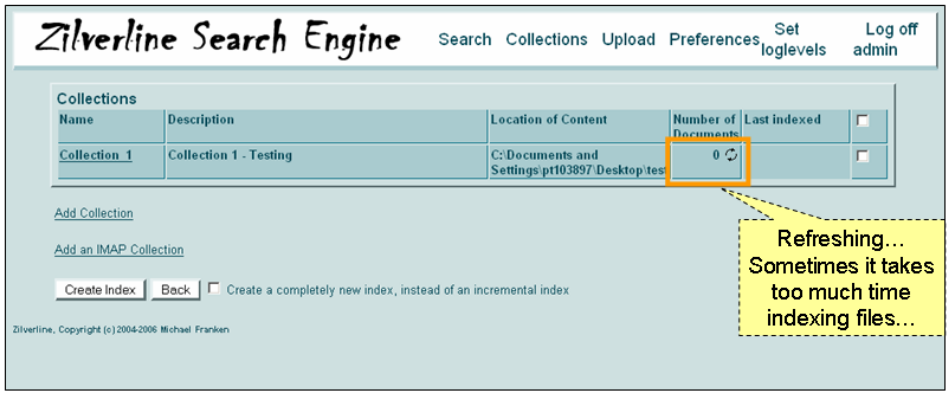
Nokia Siemens Networks

Zilverline (7)

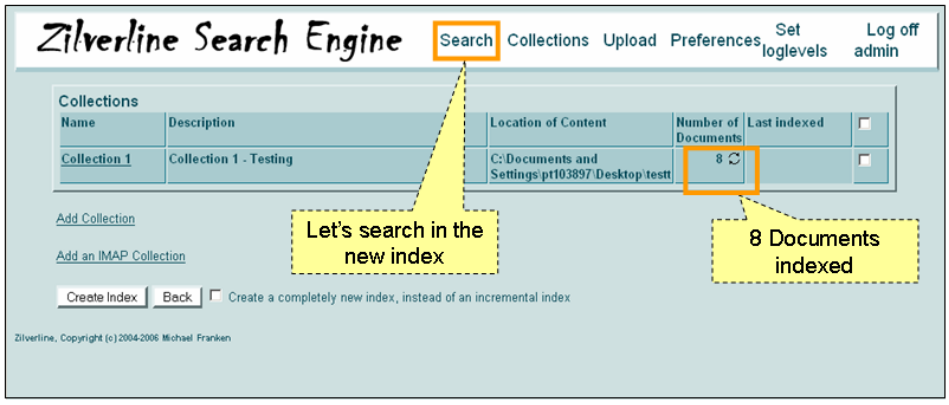


Zilverline (8)

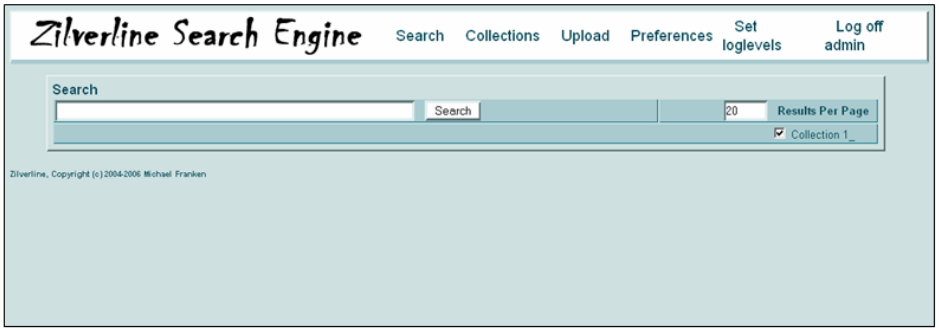
Zilverline (9)



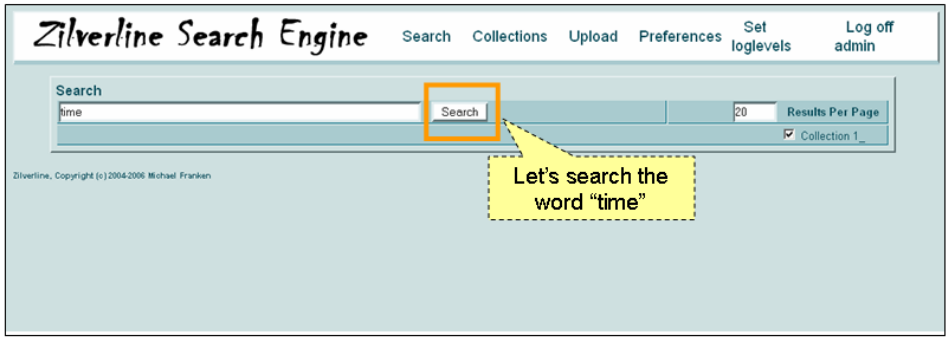
Zilverline (10)



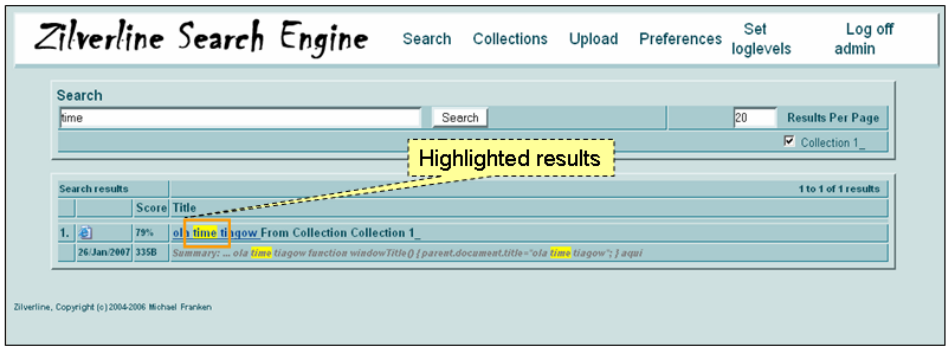
Zilverline (11)



Zilverline (12)



Zilverline (13)



Comparison

	DocSearcher	Regain	Zilverline
Indexing Time	Fast	Fastest	Slow
Indexing Rules	Flexible	More Flexible	Flexible
Indexing Creation (Difficulty)	Easy	Easiest	Easy
Filetypes Supported	HTML, PDF, RTF, TXT Word, Excel	HTML, PDF, RTF, TXT Word, Excel, Powerpoint	HTML, PDF, RTF, TXT Word, Excel, Powerpoint, ZIP, RAR
Highlighting	No	No	Yes
Several Indexes Creation	Yes	Yes	Yes
Source code complexity	Normal	High	High
Advanced Search	Yes	Yes	No
License	GPL	LGPL	Collaborative Source License

Conclusion

- The difference, in the features, between these search engines isn't too big, but only one can be chosen
- The open source search engine chosen is DocSearcher
  - It is simple to use
  - Very efficient
  - The main reason for this choice is the source code
    - Easy to understand
    - Simple architecture
    - Helpful comments
- Two features to add
  - Highlighting
  - Indexing ZIP, RAR and Powerpoint files

References

- [1] Search Tools – Enterprise Search Engines, Search Tools for Web Sites an Intranets, 1997-2007, http://www.searchtools.com, accessed January 22, 2007
- [2] DocSearcher, Overview, 2006, http://docsearcher.henschelsoft.de, accessed January 22, 2007
- [3] Jakarta POI – Java API to Access Microsoft Format Files, The Apache JakartaProject, 2002-2006, http://jakarta.apache.org/poi/, accessed January 23, 2007
- [4] PDFBox – Java PDF Library, PDFBox, 2002-2006, http://www.pdfbox.org/, accessed January 23, 2007
- [5] Regain, regain your hidden information, 2004-2005, http://regain.sourceforge.net/, accessed January 22, 2007

References (2)

- [6] Zilverline, Zilverline Search Engine, 2000-2002, http://www.zilverline.org, accessed January 22, 2007

Nokia Siemens Networks

Contacts

Tiago Honorato
IC RD1 SW2 C6 / Lisbon / Portugal
tiago.honorato@siemens.com

User eXperience Group
Siemens Networks, S.A.
Rua Irmãos Siemens, N.º1
2720-093 Alfragide
Fax: 351 – 21 416 7502

Nokia Siemens Networks