risti
Revista Ibérica de Sistemas e Tecnologias de Informação
Revista Ibérica de Sistemas y Tecnologías de Información

Recebido / Recibido: 24/10/2010
Aceitação / Aceptación: 03/12/2010

Uma proposta de Arquitetura de Software para Construção e Integração de Ambientes Virtuais de Aprendizagem

Amanda Monteiro Sizo[1], Adriano Del Pino Lino[2], Eloi Luiz Favero[1]

amandasizo@gmail.com, adrianod@ufpa.com, favero@ufpa.br

[1] Universidade Federal do Pará (UFPA) - Programa de Pós-graduação em Ciência da Computação, Caixa postal 479-66075-110, Belém-Pará, Brasil

[2] Universidade Federal do Oeste do Pará (UFOPA) – Instituto de Engenharia e Geociencias, CEP 68135-110, Santarém - Pará, Brasil

Resumo: Este artigo propõe uma arquitetura de software fundamentada no padrão arquitetural em camadas e demonstra sua aplicabilidade na construção e integração de ambientes virtuais de aprendizagem. Com o surgimento de ambientes cada vez mais modernos, sistemas legados são descontinuados pela dificuldade em mantê-los. A arquitetura proposta facilita a integração de novos módulos a estes sistemas de forma distribuída visando mantê-los competitivos. Esta proposta é implementada a partir da construção de um módulo avaliador e gerador de mapas conceituais que se integra via web service ao ambiente LabSQL. Ao fim são apresentados resultados obtidos com utilização da arquitetura e do módulo de serviço construído.

Palavras-chave: arquitetura de software; ambientes virtuais de aprendizagem; MVC; web service.

Abstract: This paper proposes a software architecture based on standard architectural and demonstrates its applicability in the construction and integration of virtual learning environments. With the emergence of more modern environments, legacy systems are discontinued by the difficulty in maintaining them. The proposed architecture facilitates the integration of new modules for these systems in a distributed way in order to keep them competitive. This proposal is implemented through the construction of a module generator and evaluator of concept maps that integrates environmental LabSQL via web service.

Keywords: software architecture; virtual learning environments, MVC, web service.

1. Introdução

Segundo a definição clássica de arquitetura apresentada por Shaw & Garlan (1996), uma arquitetura de software define o que é sistema em termos de componentes

computacionais e os relacionamentos entre estes componentes. Adicionalmente descreve a estrutura técnica, limitações e características dos componentes, bem como as interfaces entre eles. A arquitetura é o esqueleto do sistema e, por isso, torna-se o plano de mais alto-nível da construção de cada novo sistema (Krafzig, Banke & Slama, 2004).

Ao passo que aumenta a complexidade e abrangência de sistemas de informação, também aumenta a dificuldade em organizar e estruturar seus componentes (Sordi, Marinho & Nagy, 2006). Definir um padrão arquitetural é uma maneira de estimular a reutilização de componentes e padronizar a estrutura através do uso rotineiro de soluções existentes, por meio de um ponto de referência comum para as demais atividades que são executadas posteriormente a sua definição (Bass, Clements & Kazman, 2003). No caso da construção de sistemas educacionais, a arquitetura vem facilitar o desenvolvimento de sistemas que tem como características serem distribuídos e colaborativos, características estas essenciais e presente na maioria dos ambientes virtuais de aprendizagem (Costa et al, 2006).

Uma arquitetura se comporta como facilitador na construção de sistemas que derivam de uma mesma linha de produção (Queiroz & Braga, 2008), neste caso, sistemas educacionais. Para atender as peculiaridades de cada instituição de ensino, são desenvolvidas muitas ferramentas que auxiliam no ensino a distancia, como por exemplo, Teleduc, LabSQL, AulaNet, entre outros, e cada vez surgem ambiente educacionais com funcionalidade cada vez mais modernas. Nesse contexto, os sistemas legados tornam-se obsoletos rapidamente, muitas vezes desconsiderando todo o investimento, seja com pesquisa ou licença empregado na sua adoção.

Assim, este trabalho propõe uma arquitetura de *software* que facilita o desenvolvimento de ambientes de aprendizagem distribuídos que segue o padrão arquitetural em camadas, fundamentada em alguns princípios de GoF (Gamma et al., 1995), como *factory* e *abstract service*, além de padrões como business object, DAO, estabelecidos em (Alur et al., 2003). Também foram integrados nessa arquitetura os *frameworks* Jboss-Seam, EJB3 e Hibernate, que adicionalmente auxilia o trabalho do desenvolvedor e viabiliza o processo proposto neste trabalho.

A arquitetura proposta foi aplicada no desenvolvimento de um módulo que disponibiliza o serviço para geração e avaliação de mapas conceituais utilizando método desenvolvido por (Caldas & Favero, 2009) que compara via n-gramas o mapa conceitual (MC) do estudante com um modelo de resposta formada por vários mapas conceituais. Este módulo se integra ao ambiente LabSQL (Laboratório para ensino e aprendizagem de SQL), a partir de *web service* e está sendo utilizado por doze turmas de graduação e pós-graduação em diferentes áreas de atuação.

Além desta seção introdutória, este artigo está organizado como segue: na seção 2 são apresentados os trabalhos correlatos; na seção 3 é apresentada a arquitetura proposta; na seção 4 é apresentado a aplicação da arquitetura com um estudo de caso; na seção 5 é apresentado os resultados com a utilização dessa proposta e finalmente na seção 6 são realizadas as considerações finais.

2. Trabalhos correlatos

No campo educacional, tem-se o sistema AulaNet (Gerosa et al., 2004) que é um ambiente para a criação, aplicação e administração de cursos baseados na Web. Este sistema utiliza uma arquitetura implementada com técnicas de desenvolvimento baseado em componentes o que provê uma certa flexibilidade no acoplamento de funcionalidades porém precisa que as mesmas se adéqüem às restrições da plataforma J2EE.

A versão 2.0 do AulaNet utiliza uma arquitetura cliente-servidor na *World Wide Web* baseada em *servlets*, sua limitação é acessar os componentes apenas desenvolvidos em Java.

A mesma limitação acontece no Moodle, que fornece uma API – *Application Programming Interface* que implementa uma infra-estrutura de *web services* que permiti abrir o núcleo do Moodle à sistemas externos, porem limita o desenvolvedor a construir módulos em linguagem php, de acordo com seu guia próprio.

Na literatura existem vários artigos sobre as arquiteturas orientadas à serviços, nomeadamente sobre *web services* e a sua utilização na integração de sistemas (Moura & Bernardino, 2010). Essas arquiteturas possuem um problema em comum, a falta de recursos para o desenvolvimento de novos serviços, que implica diretamente na escassez de novas funcionalidades em diferentes linguagens e plataformas.

Diante desse contexto, a motivação para o desenvolvimento do trabalho é propor uma arquitetura de software simples que desenvolve eficientemente módulos de serviços de maneira interoperável, isto é, independente de plataforma ou linguagem e os integram à ambientes virtuais de aprendizado, podendo os mesmos serem reutilizados por diferentes ambientes virtuais evitando reescrita de serviços previamente concebidos.

3. Arquitetura proposta

A solução proposta apresenta uma infra-estrutura no apoio à otimização de ambientes de educação à distância através do desenvolvimento e integração de novas funcionalidades, a fim de mantê-los competitivos e evitar que esses ambientes entrem em desuso por conta do surgimento de ambientes mais modernos. Acarretando assim perda no investimento financeiro aplicado na adoção desses ambientes que por não contemplarem certas funcionalidades são descartados periodicamente.

A arquitetura aqui definida para a construção de objetos educacionais utiliza o padrão arquitetural MVC (*Model-View-Controller*), que, por definição, desacopla as camadas provendo maior flexibilidade. Esta arquitetura de software tem como principal objetivo desenvolver sistemas e integrá-los de maneira distribuída a outros ambientes independe de plataforma ou linguagem.

Alguns *frameworks* abertos foram adotados como padrões na arquitetura, como o Jboss-Seam, EJB3 e Hibernate, a reutilização de seus componentes proporcionou aumento na produtividade. A seguir, são explicadas essas tecnologias.

1. EJB3: Permite que os objetos chamados de *enterprise beans* sejam expostos como serviços *web*, de modo que seus métodos possam ser invocados por

outro aplicativo J2EE, e também por aplicativos escritos por outras linguagens de programação em diferentes plataformas (Burke & Moson-Haefel, 2007).

2. Jboss-Seam: unifica o modelo de componentes do JSF e do EJB3, além de eliminar o código de integração, o que permite que o desenvolvedor dispense mais atenção às regras de negócio (Seam, 2010).
3. Hibernate: é responsável por mapear objetos Java em tabelas de um banco de dados relacional, estreitando o *gap* conceitual que existe entre os dois, deixando o desenvolvedor livre para se concentrar no negócio da aplicação (Bauer & King, 2007).

A arquitetura e a integração com *frameworks* citados são mostrados na figura 1, através de um diagrama de componentes.

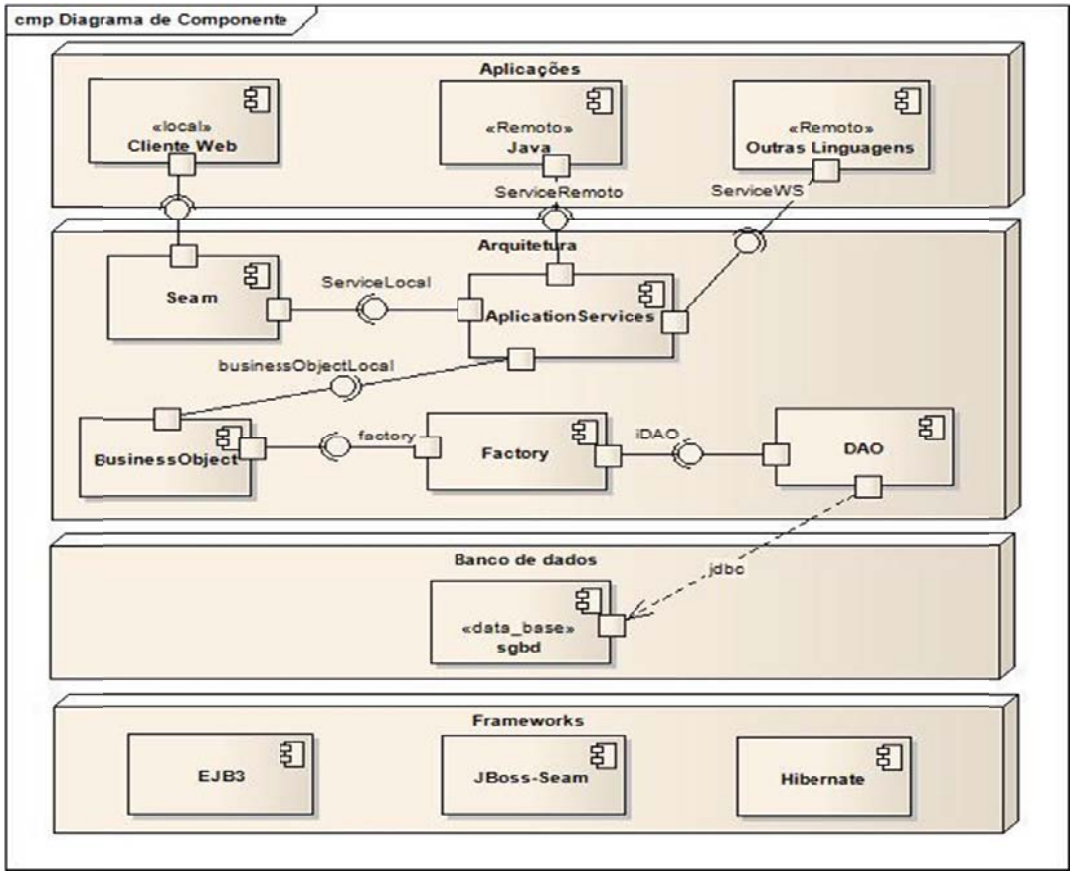

Figura 1 - Arquitetura integrada aos frameworks

3.1. Application Service

A camada *Aplication service* (Alur et al., 2003) é um padrão J2EE que centraliza a lógica de negócios de diversos componentes da camada em diversos grupos ou pontos de acesso. É a porta de entrada para efetuar uma transação. Funciona como ouvinte da camada de apresentação, ou seja, toda ação efetuada pelo usuário do sistema será "ouvida" pelo *Aplication service* e repassada para o *Business Object* executar.

A descentralização de acesso do *Aplication service*, permite que o cliente tenha acesso apenas no domínio de seu interesse, isolando assim o acesso ao restante da aplicação,

resultando em segurança e alta performance, uma vez que só é compilado o grupo de métodos referente ao serviço acessado. Diferentemente do padrão utilizado na maioria das arquiteturas, o *Facade* que sobrecarrega o processamento através de um único ponto de acesso.

3.2. Business Object

A camada *business object* é responsável por implementar a camada de lógica de negócios da aplicação e fornece os serviços para o objeto de negócio específico (Alur et al., 2003). As classes encontradas neste pacote são responsáveis por executar as transações que a aplicação cliente necessita e instanciar os componentes de persistência.

Esta camada utiliza tanto seus próprios métodos quanto os métodos disponibilizados por outros *business objects* permitindo o reuso dos serviços implementados.

3.3. Factory

Esta camada é responsável por criar objetos que devem obedecer a certos critérios dentro de uma lógica de criação complexa (Larman, 2007). Este padrão permite que a criação dos objetos DAO sejam feitos de forma transparente. Uma das vantagens de utilizar esse padrão é que ela indica qual implementação de persistência (DAO) será utilizada pela aplicação a partir de um único ponto de acesso, o que diminui o retrabalho no caso de manutenção da aplicação.

3.4. DAO

O *Data Access Object* (DAO) é um padrão conhecido na literatura como responsável por intermediar o acesso aos dados armazenados em um banco de dados (Fowler, 2006).

Cada DAO deve possuir uma interface, que especifica os métodos de manipulação de dados. Os serviços acessam apenas as interfaces dos DAOs, desconhecendo a implementação utilizada.

De uma maneira geral, objetos educacionais desenvolvidos sobre a arquitetura proposta apresentam alto desempenho, interoperabilidade além de reusabilidade, isso se deve principalmente aos seguintes fatores: descentralização no acesso, agrupamento de serviços por domínio da aplicação, e reutilização de implementação dos DAOS.

4. Aplicação da Arquitetura

Com a difusão dos ambientes virtuais de aprendizagem é cada dia mais relevante o estudo e desenvolvimento de ferramentas que possam avaliar os aprendizes à medida que diferentes formas de ensino são difundidas. Entende-se ser necessário saber tirar proveito das possibilidades tecnológicas que emergem com esses novos espaços pedagógicos a fim de otimizar os ambientes através da inclusão de novas abordagens (Behar & Leite, 2005).

Mapas Conceituais têm a sua origem no movimento da teoria construtivista da aprendizagem, de Ausubel (1980) e são utilizados como ferramenta de apoio à

representação de conhecimento, com um vasto relato de aplicações em educação, notadamente no apoio à avaliação da aprendizagem (Araújo,Menezes & Cury, 2002).

O LabSQL é uma ferramenta de ensino e aprendizagem da linguagem SQL e sua proposta inicial está centrada na avaliação automática de consultas SQL (Lino et al., 2007). Atualmente, o LabSQL agrega funcionalidades que são resultados de várias pesquisas.

Dando continuidade às pesquisas que tem como fim maximizar a aprendizagem dos estudantes, pretende-se agregar os benefícios pedagógicos alcançados por meio do ensino a partir da utilização de mapas conceituais, a fim de detectar através do mapeamento dos conceitos chaves a deficiência dos estudantes na área de banco de dados.

4.1. Estudo de Caso

Para compor o estudo de caso desta pesquisa, foi desenvolvido um módulo de serviço para geração e avaliação de mapas conceituais, que são implementações de *webservices* disponibilizadas para o LabSQl conforme mostra a figura 2. As telas aqui mostradas fazem parte do LabSQL e acessam o serviço disponibilizado pelo módulo proposto.

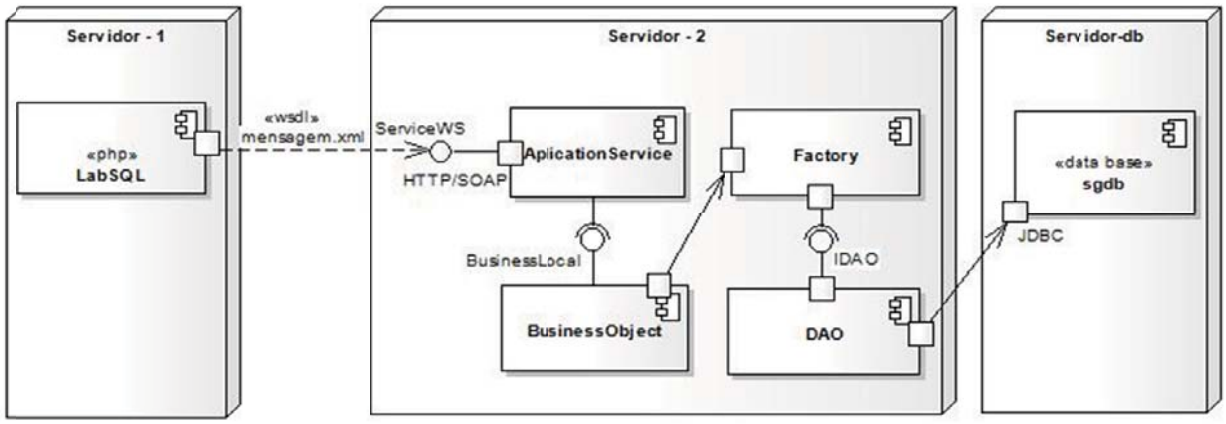

Figura 2 - Diagrama de implantação - Integração entre as aplicações via web service

A primeira fase para construir um projeto baseado na arquitetura, é definir as entidades decorrentes do modelo de negócio. Essas entidades compõem a camada *entity* da arquitetura, isto é, entidades do domínio da aplicação também conhecidas como POJO(*Plain Old Java Objects*) Para o projeto de estudo de caso foi definido o modelo do domínio do negócio como mostrado na figura 3 a seguir:

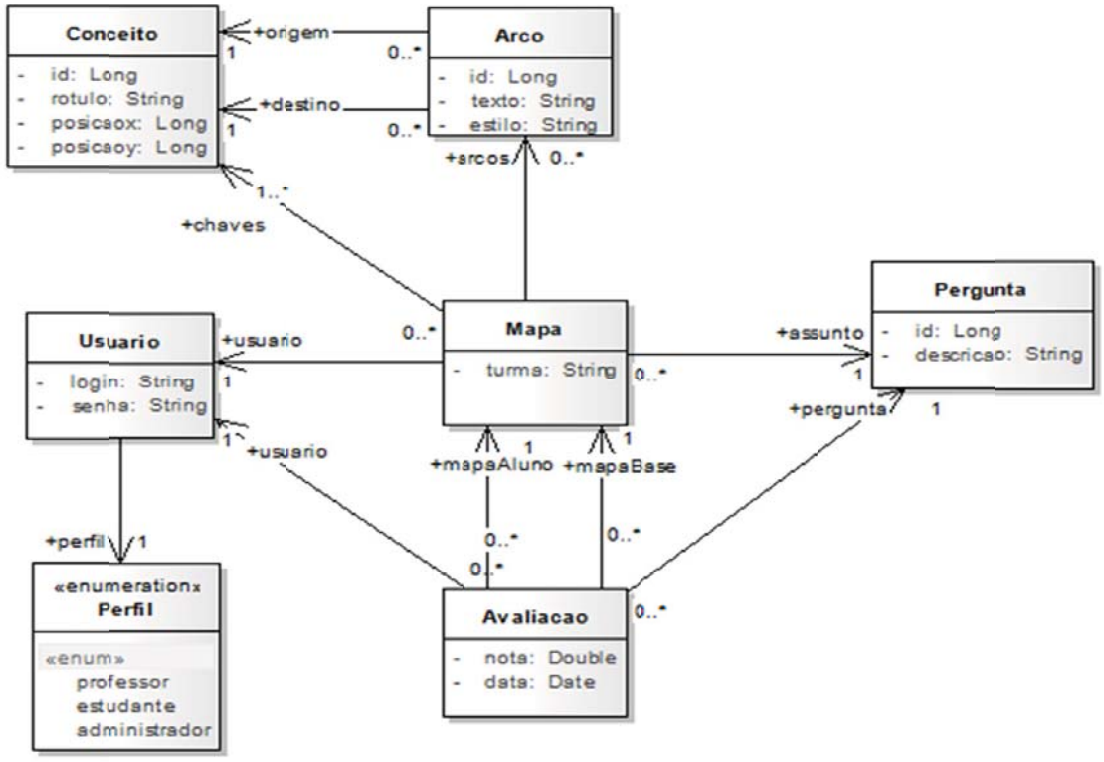

Figura 3 - Diagrama de classes – Modelo de Negócios

O módulo para geração e avaliação de mapas conceituais possui como requisitos principais: a) manter mapa conceitual: Esta funcionalidade permite que usuários com perfil de professor ou estudante gerem mapas conceituais, porem só o professor pode cadastrar seu mapa como modelo, isto é, o mapa que serve de referência na correção dos mapas dos estudantes e b) corrigir mapa conceitual: Esta funcionalidade compara o MC do estudante com o MC base previamente cadastrado pelo professor, e compara com os mapas conceituais dos demais estudantes gerando uma nota de acordo com seu número de acertos. Caso o aluno acrescente um conceito inválido o mesmo é diferenciado.

O componente desenvolvido tem como propósito principal a avaliação automática de MCs a partir da análise de similaridade dos MCs dos estudantes contra um modelo de resposta do professor, retornando um escore da avaliação quantitativa. Este resultado é obtido através da análise via bigrama da formação conceito-link ou link-conceito e análise via trigrama da formação dos arcos conceito-link-conceito ou link-conceito-link (Caldas & Favero, 2009).

A figura 4 mostra a tela de resultado da avaliação de mapas conceituais produzida pelo estudante, como pode ser observado, foi gerada uma nota para questão 217 que define o conceito de banco de dados relacional, sendo que além de apresentar o histórico de notas para a questão o sistema também aponta o conceito que não faz parte da definição.

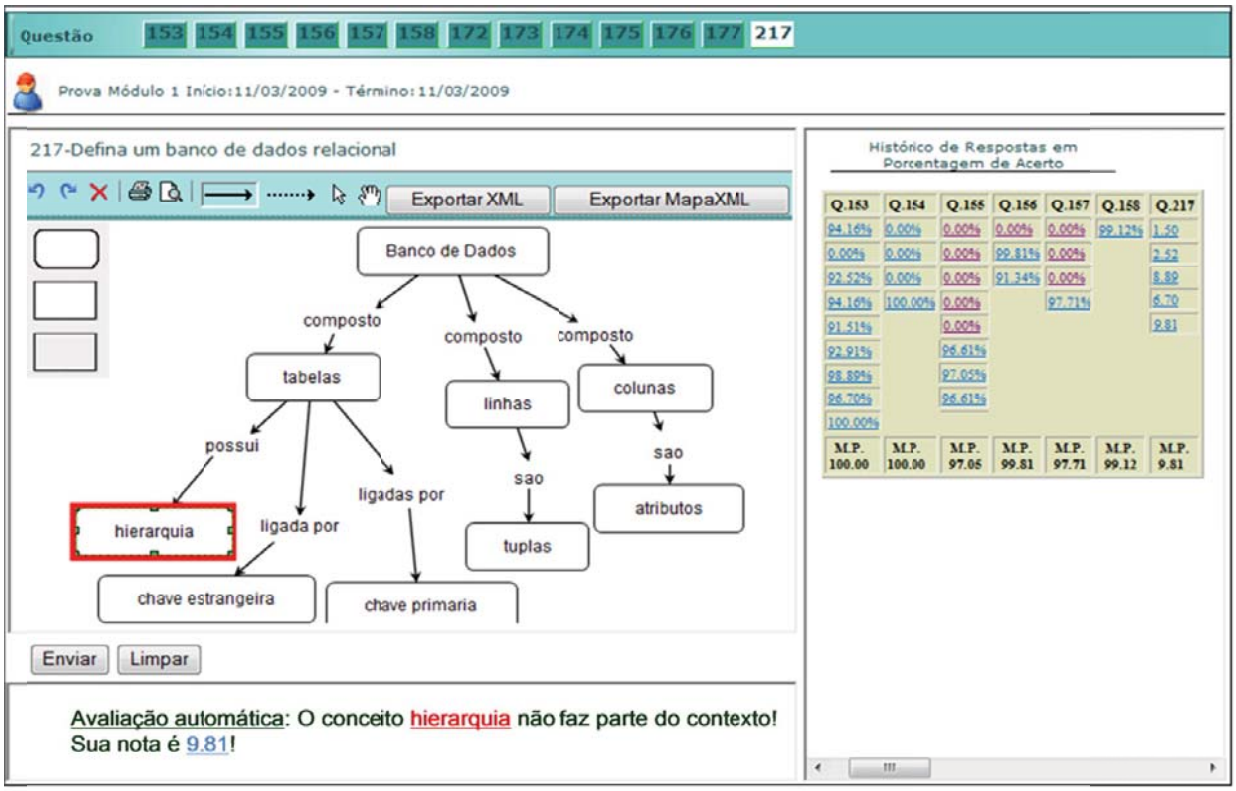

Figura 4 - Resultado da avaliação do mapa conceitual do aluno

Os serviços como, cadastrar mapas conceituais, avaliar mapas conceituais entre outros do módulo citado, são disponibilizados por uma interface webservice (ServiceWS), essa interface é acessada pelo LabSQL que consegue assim fornecer esses serviços a partir de sua aplicação a seus usuários finais.

A Figura 5 ilustra o fluxo de execução para a funcionalidade cadastrar mapas, observe que cada camada possui uma interface (ServiceWS, BusinessMapaLocal e IDAOMapa) que define quais serviços estão disponibilizados e sua respectiva implementação (AplicationService, BusinessMapatImpl, IDAOMapaImpl) com exceção da camada *factory* que já disponibiliza seus métodos públicos na própria classe.

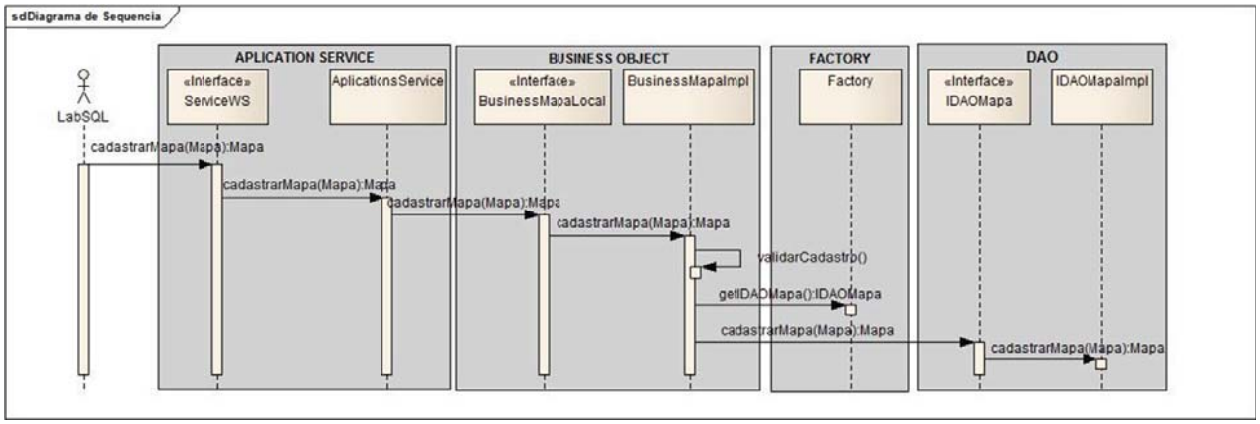

Figura 5 - Diagrama de seqüência - Método cadastrar mapa

Esta arquitetura além de mais simples adiciona performance no tratamento de seus objetos, isso se deve aos seguintes fatores:

1. para cada funcionalidade disponibilizada existe um *Aplication Service*, evitando a centralização em um único ponto de acesso e em conseqüência sua sobrecarga, resultando na diminuição no tempo de resposta no processamento das operações;
2. inclusão da camada de *businesse object* que além de ser responsáveis pelas regras de negocio, permite que os serviços implementados em um BO sejam reutilizados em outros BO permitindo um relacionamento cíclico entre os mesmos, evitando assim reescrita de código de uma mesma função em BOs distintos;
3. inclusão da camada *factory* que permite a partir de um único ponto indicar qual DAO será utilizado na aplicação, isso aumenta a produtividade na fase de desenvolvimento e manutenção. Em outras arquiteturas se houver mudanças na implementação no DAO a ser utilizado, o desenvolvedor teria que apontar em cada BO que utiliza o DAO a nova implementação, ocasionando aumento de retrabalho. Esta camada também possui como vantagem a reutilização da instância do DAO que esta na memória auxiliando na melhoria do desempenho.

5. Resultados Obtidos

5.1. Quanto à arquitetura

A definição de padrões de qualidade é importante na elaboração de produtos de software, notadamente, na construção de componentes (Simão & Belchior, 2002). Os padrões são usados como parâmetros, para dar mais atenção aos aspectos que levem os componentes a níveis de qualidade esperados, com maior rapidez e menor custo. Alguns trabalhos abordam mais diretamente as características de qualidade para componentes de software (Preiss et al., 2001; Bertoa, 2002).

Entretanto, ainda se faz necessário um conjunto de características específicas, que possa ser usado para avaliar vários aspectos que definem uma arquitetura com qualidade. As características aqui avaliadas foram organizadas segundo a norma ISO/IEC 9126 (ISO, 2001), que define requisitos de qualidade para componentes de software, além de ser um padrão estabelecido internacionalmente.

A Tabela 1 apresenta o conjunto das características de qualidade organizado segundo o modelo de qualidade da ISO/IEC 9126, e os principais fatores que evidenciam o atendimento desse requisito na arquitetura proposta.

Tabela 1 - Requisitos de qualidade atendidos

Característica	Definição	Atendimento
Funcionalidade	Característica do componente prover serviços (funções) que satisfaçam as necessidades especificadas, quando o componente for usado sob condições específicas (ISO, 2001)	A arquitetura desenvolveu o módulo de serviço satisfazendo as especificações determinadas pelo domínio do negocio de mapas conceituais.
Confiabilidade	Característica do componente que o faz manter em um determinado nível de desempenho, quando usado sob condições especificadas (ISO, 2001).	A camada de aplication Service descentralizou o acesso, diminuindo o tempo de resposta para o usuário.
Usabilidade	Característica do componente ser entendido, aprendido, usado e atrativo para o usuário, quando usado sob determinadas condições (ISO, 2001).	As camadas são fracamente acopladas e fornecem uma maneira eficiente e padronizado de desenvolvimento ao arquiteto de software.
Eficiência	Característica do componente realizar suas funções sem desperdício de recursos, sob condições especificadas (ISO, 2001).	A camada Business utiliza tanto seus próprios métodos quanto os métodos disponibilizados por outros business objects permitindo o reuso dos serviços implementados.
Manutenibilidade	Característica do componente ser modificado. Modificações podem incluir correção, melhorias ou adaptação do componente para mudanças em ambiente, e em requisitos e especificações funcionais (ISO, 2001).	A camada Factory indica qual implementação de persistência (DAO) será utilizada pela aplicação a partir de um único ponto de acesso, o que diminui o retrabalho no caso de manutenção da aplicação.
Portabilidade	Característica do componente ser transferido de um ambiente para outro. O ambiente pode ser uma organização, um hardware, um ambiente de software ou uma aplicação. (ISO, 2001)	As tecnologias e frameworks especificados proporcionam a construção de serviços independentes de plataforma ou linguagem.

Importante salientar que, na análise apresentada não pretende-se contemplar os requisitos da ISO/IEC 9126 em sua totalidade, pois uma característica pode ser observada em vários aspectos da arquitetura e por uma questão de organização , uma determinada característica é definida somente uma única vez, onde se observa que a sua maior influência é exercida.

5.2. Quanto à aprendizagem

A versão do LabSQL que contém a funcionalidade dos Mapas Conceituais foi utilizada por várias turmas diferentes e os resultados obtidos até o momento são satisfatórios. Essa versão foi testada por 12 (doze) turmas, nos níveis de graduação e pós-graduação da UFPA, durante o período de 4 (quatro) semestres letivos. A partir dos dados obtidos destacamos as seguintes informações: melhora no desempenho, aumento na execução de exemplos e aumento de submissões de respostas em exercícios.

Confrontando o histórico de notas das turmas sem MC no período de 2004 até 2007 contra o histórico de 2008 até 2009, no nível de graduação, observou-se que, nas turmas tradicionais sem a utilização do MC, o desempenho é menor do que nas turmas com a utilização do MC. O aspecto positivo da utilização de MC quanto aos resultados

pode ser observado por meio do acréscimo de 13% para 27% no percentual de alunos com conceito de bom a excelente (Figura 6 - A).

Quanto à reutilização do sistema ocorre quando um aprendiz utiliza o ambiente mais de uma vez a fim de treinar ou exercitar exemplos práticos que estão no LabSQL, essa medida é extraída automaticamente do ambiente pode ser utilizada como complemento na avaliação do aprendizado do estudante (Lobato et al., 2008). Em Silva et al. (2008) o LabSQL não possuía o MC, e em sua pesquisa foi relatada que o número de exemplos realizados no sistema era de 1.706 submissões, contra um total de 2.527 submissões na versão atual que contém o MC, conforme gráfico (Figura 6 - B).

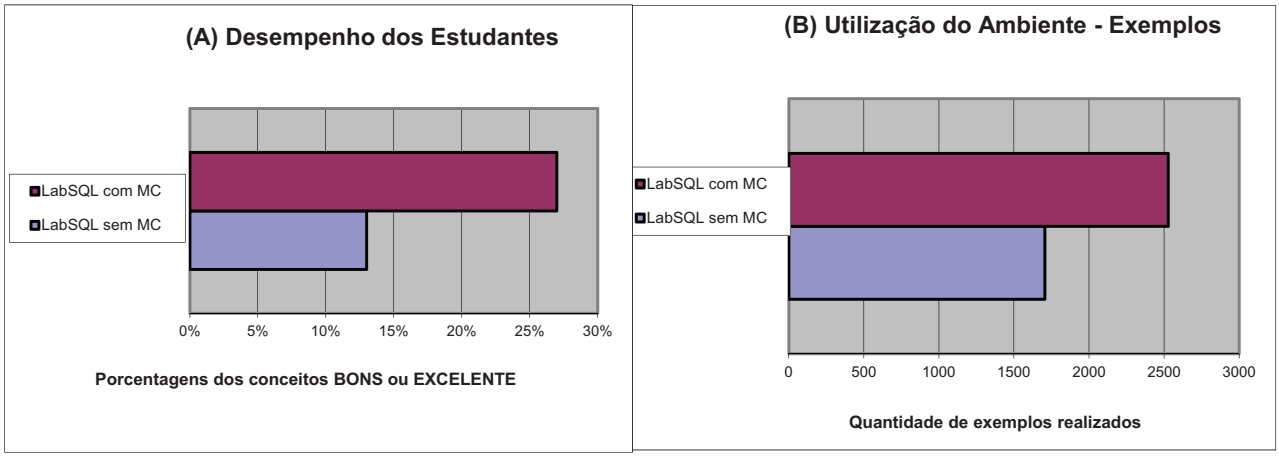

Figura 6 - Gráficos de Desempenho e Reutilização do sistema

Além disso, uma consulta no banco de dados do histórico do perfil dos aprendizes revelou que ocorreu um acréscimo de utilização dos módulos de treinamentos e execução de exercícios. Acreditamos que o MC trouxe o benefício da reflexão do aprendiz e na melhora da qualidade das respostas submetido pelo aprendiz, isso pode ser constatado por meio de uma consulta que confronta os conceitos finais dos aprendizes nos períodos informados. Em média foram acrescidos 12% no conceito final dos estudantes e uma redução de 6% no índice de reprovação. A Figura 7, ilustra que ocorreu uma redução nesses índices de reprovação, que comprovam que o LabSQL com MC aumentou sua usabilidade.

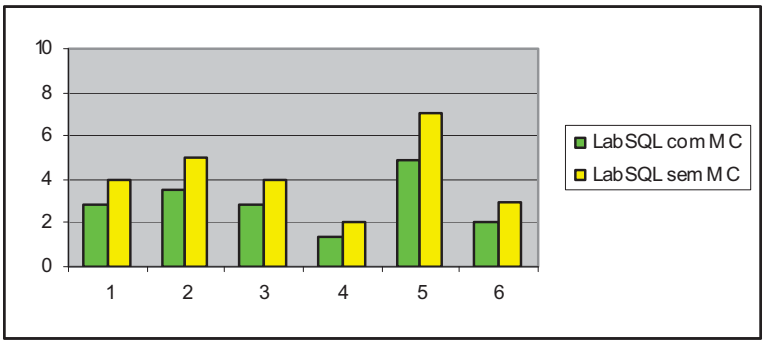

Figura 7- Gráfico de índice de redução de reprovação

6. Conclusão

Em termos gerais, este artigo apresentou uma arquitetura que integra aplicações distribuídas de maneira eficiente além de potencializar o processamento durante a execução da aplicação. Também contemplou requisitos, que de acordo com (Sordi , Marinho & Nagy, 2006) fazem parte de uma boa arquitetura, como: desempenho, reusabilidade, manutenibilidade, entre outros.

O grande diferencial desta arquitetura é o aumento de desempenho e reusabilidade proporcionada por meio do acesso descentralizado aos serviços na *camada business object*. Também prove a interoperabilidade que desempenha um papel essencial em qualquer infra-estrutura onde os atores e suas aplicações são distribuídos e heterogêneos.

Como principal contribuição dessa arquitetura tem-se o acoplamento eficientemente de serviços independente de plataforma ou linguagem em qualquer ambiente virtual de aprendizagem, otimizando os mesmos. Além disso, a possibilidade de inserção de novas funcionalidades aos ambientes trouxe flexibilidade, ou seja, pode-se criar se desejável, módulos dentro do próprio ambiente que se integram a outros ambientes virtuais ou sistemas administrativos, requisito este muito desejável em qualquer ambiente educacional.

A proposta foi viabilizada por meio do desenvolvimento de um componente que gera e avalia mapas conceituais que se integra ao ambiente LabSQL através de webservices.

Como perspectiva futura, pretende-se estender o experimento à integração de novas funcionalidades a outros ambientes virtuais de aprendizagem, também se pode agregar novos recursos a fim de contemplar funcionalidades administrativas ao ambiente como administrar notas, faltas, cadastro de usuários etc.

Referências

Alur, D.; Malks, D.; & Crupli, J.(2003). Core J2EE Patterns: Best Practices and Design Strategies, *Second Edition*. *Pearson*.

Araújo, A. M.T., Menezes, C. S. & Cury, D. (2002). Um Ambiente Integrado para Apoiar a Avaliação da Aprendizagem Baseado em Mapas Conceituais. *Anais do XIII SBIE,* São Leopoldo, RS.

Ausubel, D. (1980). Psicologia Educacional. - 2. Ed. Interamericana.

Bass, Len; Clements, P., Kazman, R. (2003). Software Architecture in Practice. *Addision-Wesley*, 2. ed.

Bauer, C. & King G. (2007). Java Persitence com Hibernate. *Ciência Moderna*.

Behar, P.A. & Leite, S.M. (2005). Criando novos espaços pedagógicos na internet: O ambiente ROODA.*Revista de Gestão da Tecnologia e Sistemas de Informação*.

Bertoa, M. F., Troya, J.M. & Vallecillo, A. (2002). Aspectos de la Calidad en el Desarrollo de Software Basado em Componentes, *Capítulo do livro: Calidad en el desarrollo y mantenimiento del software.*

Burke, B. & Moson-Haefel, R. (2007). Enterprise javaBeans 3.0. 5th. *Peason*.

Caldas, V.M. & Favero, E.L. (2009). Uma Proposta de Avaliação Automática de Mapas Conceituais para Ambientes de Ensino a Distância*", XXXV Conferencia Latinoamericana de Informática* – Pelotas – RS.

Costa, K. C. F., Harb, M. D. P. A., Brito, S. R. & Favero, E. L.(2006). Um sistema de recomendação para ambientes virtuais de aprendizagem baseado em agentes e componentes de software. *32º Conferencia LatinoAmericana de Informática*- Santiago – Chile.

Fowler, M. (2006). Padrões de Arquitetura de Aplicações Corporativas. *Bookman*.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J.(1995). Design patterns: Elements of reusable object oriented software. *Reading*,

Gerosa, M. A., Raposo, A.B., Fulks, H. & Lucena, C.J.P. (2004).Uma arquitetura para o Desenvolvimento de Ferramentas Colaborativas para o Ambiente de Aprendizagem AulaNet, XV Simpósio Brasileiro de Informática na Educação

ISO/IEC 9126-1.(2001). Information Technology . Software Product Quality. part 1.

Krafzig, D., Banke, K. & Slama, D. (2004).Enterprise SOA: Service-Oriented Architecture Best Practices. Indianapolis: Prentice Hall.

Larman, C. (2007). Utilizando UML e Padrões: uma introdução à análise e ao projeto orientado a objetos. 3ª edição, Porto Alegre: Bookman.

Lino, A. D. P., Silva, A. S., Favero, E. L. & Harb, M. D. P. A. (2007). Avaliação automática de consultas SQL em ambiente virtual de ensino-aprendizagem. *2ª Conferência Ibérica de Sistemas e Tecnologias de Informação*, v.1. p.89 – 100.

Lobato, A. S., Harb, M. D. P. A., Lino, A. D. P., Elói L. F., Silva, H. A. N & Santos, T. L. T. (2008). Aplicando Rubrica para Avaliar Qualitativamente o Estudante no LabSQL. *Conferencia LatinoAmericana de Informática* , Santa Fe.

Moura, R. & Bernadino J. (2010). Um modelo para a integração de serviços – Moodle e Sistemas de Gestão Acadêmica. *Revista Ibérica de Sistemas de Tecnologia da Informação*, N◦ 5, pag. 31.

Preiss, O., Wegmann, A. & Wong, J. (2001). On Quality Attribute Based Software Engineering. *27th Euromicro Conference*. Warsaw, Poland, September.

Queiroz, P.G. & Braga, R.T.V. (2008). Desenvolvimento de Linhas de Produtos de Software com uma Arquitetura Orientada a Serviços. *13º Workshop de Teses e Dissertações em Engenharia de Software*.

Seam. (2010). Sitio oficial do framework seam. Disponível em: http://seamframework.org/Documentation/SeamDocumentation#HSeamReferenceDocumentation, Acesso: 21 out 2010.

Shaw, M. & Garlan, D.(1996). Software Architecture. Perspectives on an Emerging Discipline. *Prentice Hall*.

Silva, H. A. N., Lino, A. D. P., Silveira, A. M., Favero, E. L., Santos, T. L. T. & Morais, R. F. (2008). Um sistema baseado na lógica difusa para decidir os conceitos finais dos estudantes críticos. *Workshop sobre Educação em Computação* | SBC - Belém. p.1 – 10

Simão, R.P.S. & Belchior, A.D.(2002). Um Padrão de Qualidade para Componentes de Software. *I Simpósio Brasileiro em Qualidade de Software*. Gramado, RS.

Sordi, J.O., Marinho, B.L. & Nagy, M.(2006). Benefícios da Arquitetura de Software Orientada a Serviços para as Empresas: Análise da Experiência do ABN AMRO Brasil. Revista de Gestão da Tecnologia e Sistemas de Informação.