JLearningServices: Um Framework para Serviços Síncronos em Ambientes para EAD

Letícia Rafaela Rheinheimer

leticia@cs.inf.br

Sérgio Crespo Coelho da Silva Pinto

crespo@exatas.unisinos.br

Universidade do Vale do Rio dos Sinos - UNISINOS

Centro de Ciências Exatas e Tecnológicas

Av. Unisinos 950, 93022-000 São Leopoldo, RS, Brasil

Abstract

Internet popularized the use of synchronous communication services. Many of these services are being used in Web-based education environments. Education area is very dynamic, so the use of Software Engineering technologies becomes necessary for fast prototype development and computer systems generation. This paper presents the development of an educational Framework called JLearningServices. This tool must provide sinchronous services generation for one-to-one, one-to-many and many-to-many communication in distance learning, applying Frameworks technology. Design Patterns are used in the prototype design to provide a better representation of its hot-spots and frozen-spots.

Keywords: *Frameworks, Design Patterns, Web-based education*.

Resumo

A Internet popularizou a utilização de serviços de comunicação síncrona, sendo que muitos destes estão sendo utilizados como ferramentas de comunicação em ambientes para educação baseada na *Web*. Em virtude do campo educacional ser muito dinâmico, torna-se necessário a utilização de tecnologias de Engenharia de Software para a prototipação e geração rápida de sistemas computacionais. JLearningServices utiliza a tecnologia de *Frameworks* para a geração de serviços síncronos para comunicação um-para-um, um-para-muitos e muitos-para-muitos, dirigidos a ambiente virtuais de aprendizagem baseados na *Web*. JLearningServices utiliza um conjunto de *Design Patterns* em seu modelo de forma a proporcionar uma melhor representação dos seus vários *hot-spots* e *frozen-spots*.

Palavras-chave: *Frameworks*, *Design Patterns*, informática e educação.

1 - Introdução

A educação/treinamento baseada na *Web* usa a WWW como meio para a publicação do material didático, aplicação de tutoriais, aplicação de provas e testes e comunicação com estudantes. Também compreende o processo de uso da *Web* como veículo de comunicação para a apresentação de aulas à distância (conferência multimídia). As tecnologias de educação/treinamento baseada na *Web* estão em pleno uso, utilizando uma série de ferramentas que auxiliam os processos de cooperação, coordenação e comunicação [17] e [6].

A utilização de ferramentas que possibilitem a cooperação entre os aprendizes torna-se cada vez mais utilizadas nos ambientes virtuais. Estas podem ser tanto síncronas (*chats*, ICQ, dentre outras) como assíncrona (*email*, listas, fórum, etc.). Estas ferramentas, em sua maioria, já existem bem antes do surgimento dos ambientes que oferecem suporte a educação baseada na *Web* (EAD). A maioria dos ambientes virtuais utilizam, para possibilitar a cooperação e comunicação entre os aprendizes, artefatos de *software* que não foram projetados com enfoque educacional. Pôde-se perceber, ao conhecer alguns ambientes no decorrer deste trabalho, que ferramentas como *chat*, *email*, etc. são oferecidas como recursos integrados aos ambientes, mas sem grande relevância no contexto educacional: são apenas auxiliares, quando poderiam contribuir de maneira mais específica no processo de ensino/aprendizagem. Existe a integração/adaptação de ferramentas já prontas (*chat*, por exemplo), em lugar de se fazer um estudo na área e desenvolver algo novo, com características específicas para o ensino.

Existe uma grande preocupação com a qualidade nesta área, em diversos sentidos, inclusive no que diz respeito a ferramentas que auxiliem o professor e que propiciem uma melhor interação entre os alunos participantes desse processo de ensino/aprendizagem [17]. Assim, vêm-se buscando novas formas de suprir essas necessidades e dar ao EAD maior qualidade, segurança e confiabilidade.

Este trabalho vem trazer uma contribuição neste sentido, oferecendo ferramentas de serviços síncronos em seus diferentes tipos de comunicação (um-para-um, um-para-muitos e muitos-para-muitos) voltadas para a área do ensino.

Este artigo apresenta na seção 2 uma breve descrição da tecnologia utilizada para a implementação do *Framework* JLearningServices, que possibilita a geração de aplicações síncronas para ambientes de EAD.

Na seção 3, o artigo apresenta um estudo sobre sistemas de comunicação síncrona, em diversos ambientes na *Web*, de forma a permitir elicitar os principais requisitos para a definição do *Framework.*

Na seção 4, será descrito o *Framework* JLearningServices, enfatizando o seu modelo de classes e objetos, os seus *hot-spots*, *kernel* e possibilidade de instanciação.

Na seção 5, apresenta-se a conclusão e os trabalhos futuros.

As referências bibliográficas são apresentadas na seção 6.

2 - *Design Patterns* e *Frameworks*

Com o advento da *Web*, uma nova plataforma se tornou disponível, introduzindo uma nova forma de disponibilizar a informação para consulta. A velocidade com que surgem novas necessidades provoca um constante repensar sobre a forma de se desenvolver sistemas de *software*.

Diversas metodologias são projetadas de forma a atender a estas necessidades; porém, dois aspectos chamam mais atenção: a tecnologia a ser utilizada para o desenvolvimento de *software* e a possibilidade da reutilização de rotinas, bibliotecas e serviços já desenvolvidos.

Segundo Umar [22], as novas tecnologias para o desenvolvimento de aplicações têm-se centrado em: uso da *Web* e aplicações Cliente/Servidor. Mattsson [19] e Bassett [1] colocam questões importantes sobre o processo de desenvolvimento de *software* como:

- – Como construir novas aplicações usufruindo das vantagens deste novo paradigma?
- – Que tipo de estrutura é necessária para dar suporte à reutilização e ao desenvolvimento rápido e confiável de sistemas de *software*?

Pesquisas têm mostrado que altos níveis de reutilização de *software* podem ser alcançados através do uso de *Frameworks* orientados a objetos e *Design Patterns* [4], [20] e [8].

Segundo Pree, *Frameworks* unificam sistemas de *software* para um domínio específico e constituem em uma construção semi-acabada de blocos de códigos prontos para uso; em associação com uma arquitetura global, obtida pela composição e interação entre os blocos. Os aspectos de um *Framework* que não são projetados para adaptação (*frozen-spots*) são representados por meio de métodos *template* e constituem o *kernel* do *Framework* [20]. Um *hot-spot* é uma classe abstrata que não possui implementação e deve ser especializada (customizada) para necessidades específicas da aplicação a ser gerada. A especialização pode ser realizada tanto por herança como por delegação, dependendo da forma como os *hot-spots* foram planejados [4].

Um *Framework* orientado a objeto captura os aspectos comuns à uma família de aplicações. Ele também captura a experiência do projetista durante o processo de construção da aplicação. Isto possibilita reutilização tanto em *design* como em programação [21].

Pesquisas atuais reconhecem a necessidade da integração de *Frameworks* e *Design Patterns*, uma vez que os mesmos também promovem flexibilização e reutilização. Porém, não está claro como identificar os *hot-spots* em *Frameworks*, e como fazer a conexão entre a seleção do *Design Pattern* e a implementação destas variações [4].

Reutilizar *Frameworks* muitas vezes significa adaptar alguns componentes e ou blocos para uma necessidade específica sobrepondo alguns dos métodos de suas subclasses.

A seguir, será apresentado um estudo das funcionalidades de alguns sistemas de comunicação síncrona obtendo uma melhor elicitação dos requisitos para a modelagem do JLearningServices.

3 - Análise de sistemas de comunicação síncrona em ambientes de EAD.

Diversos sistemas de comunicação síncrona foram observados com o objetivo de se fazer uma engenharia reversa e, a partir disso, definir as características pertinentes ao *kernel* do *Framework* e aquelas que deveriam constituir os *hot-spots*. Para exemplificar esse processo, serão utilizados os seguintes sistemas: *Chat* do Colégio Santa Úrsula (www.ursula.com.br/colegio/chat), Sala de *chat* do LEC – UFRGS (www.psicologia.ufrgs.br), ICQ (www.mirabilis.com), *Instant Messenger* (www.aol.com/aim/home.html) e *The Palace* (www.humanas.unisinos.br/ambiente/oficinas/the_palace/index.htm).

No *Chat* do Colégio Santa Úrsula, foram observadas as seguintes características: o sistema possui um servidor que permite o acesso de diversos clientes. Cada cliente possui apenas uma sala de bate-papo, que permite a interação entre diversos participantes. Para entrar na sala é necessário *login* e permite-se conversa reservada. Não existe a possibilidade de se formatar o texto das mensagens a serem enviadas.

Da mesma forma, a Sala de *chat* do LEC – UFRGS possui um servidor que permite acesso de diversos clientes e várias salas são oferecidas. Há uma lista de ações e uma lista de sons que podem ser utilizados. A mensagem a ser enviada pode ser formatada. É permitido a realização de conversa reservada bem como obter o *profile* de um usuário.

O ICQ, no momento da instalação, permite que o usuário se conecte a um dos servidores disponíveis na Internet, define para o usuário um número de identificação e permite que informe outros dados como o nome, por exemplo. Pode-se escolher o tipo de comunicação desejado: *chat*, voz, *message board*, conferência (múltiplos usuários), transferência de arquivos, jogos. Serve de plataforma para aplicações *peer-to-peer*, como *Netscape CoolTalk* e *Microsoft NetMeeting*.

No *Instant Messenger*, o usuário possui uma lista de contatos e indica, através de texto, o seu *status* (*online* ou *offline*). O *Instant Messenger* permite que se localize outros usuários com interesses em comum.

The Palace possui diversos provedores. Cada servidor permite o acesso de diversos clientes, onde vários participantes interagem. A conversação pode ser feita através de texto, imagens,

som. Existem ferramentas para desenhar na tela. É possível enviar mensagens para todos ou para algum participante específico.

É possível perceber pontos em comum nos sistemas síncronos descritos, bem como características que não estão presentes em todos eles, ou que se apresentam em alguns de formas diferentes. Esses pontos em comum devem fazer parte do *kernel* do *Framework*, enquanto os *hot-spots* serão consituídos pelas características diferenciadas. O processo de engenharia reversa realizado resultou no desenvolvimento de um primeiro modelo, que serviu de base para a modelagem do *Framework*. Esse modelo pode ser visto na **Figura 1**.

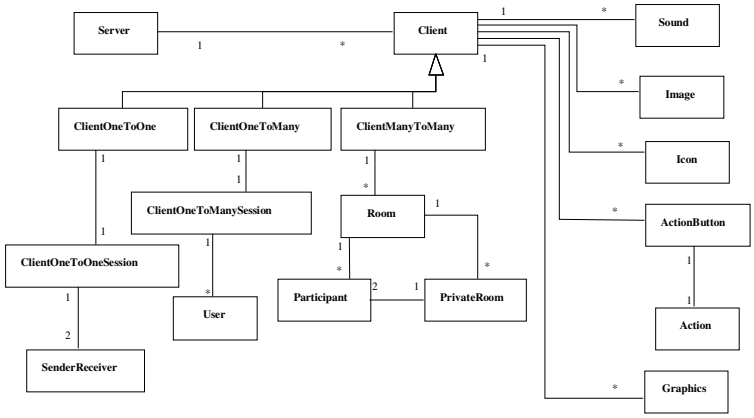

Figura Erro! Indicador não definido. **- Modelo genérico dos serviços síncronos analisados.**

Os sistemas descritos constituem apenas algumas das ferramentas síncronas analisadas, utilizadas neste artigo para melhor ilustrar o processo de engenharia reversa e definição dos requisitos. Outras ferramentas síncronas, em seus diversos tipos, foram observadas a fim de se obter os requisitos necessários para o desenvolvimento do *Framework*.

3.1 - Pontos de flexibilização dos ambientes analisados

Com base nas informações obtidas, foi possível definir as classes do *kernel* e dos *hot-spots* do *Framework*. Foram definidos alguns *hot-spots* próprios de uma aplicação muitos-para-muitos e outros que são comuns a todos os tipos de aplicação.

Como pontos a serem flexibilizados de um sistema de comunicação síncrona, temos:

- – Definição de papéis para os participantes - para cada participante pode ser definido um papel (aluno, professor) que o permita realizar certas operações tais como:
    - o Existência de um mediador na sala - pode-se definir o papel de um mediador, que teria controle sobre os demais participantes, ou não ter essa figura presente,
    - o Execução de aplicações em outros clientes*:* um professor pode, por exemplo, mandar abrir uma URL na máquina dos alunos e
    - o Possibilidade de conversa reservada – permitir que dois usuários troquem mensagens em uma seção de *chat* sem que outros possam ler.
- – Definição de funcionalidade - a cada papel criado pode ser relacionada uma série de funcionalidades como, por exemplo:
    - o permissão para retirar um participante de uma sala ,
    - o ativação de uma URL em todas as aplicações clientes,
    - o uso de recursos gráficos - Uma aplicação pode ter ícones para identificar os usuários ou utilizar botões com ações definidas. O usuário pode fazer desenhos e enviá-los,

- formatação de mensagem - O texto de uma mensagem a ser enviada pode ter cor, tamanho da fonte e estilo definidos pelo usuário e
- uso de som.

A seguir será apresentado o modelo de classes do *Framework* proposto levando-se em consideração os aspectos já elicitados dos diversos sistemas de comunicação síncrona.

4 - JLearningServices: Um *Framework* para serviços síncronos em ambientes educacionais baseados na Web.

A geração de ferramentas para suporte a educação a distância tem-se mostrado promissora, embora alguma destas não incorporem os requisitos necessários para o enfoque educacional. Em virtude disto, faz-se necessário a obtenção de maior flexibilidade no processo de geração de artefatos de *software* que promovam suporte a EAD. O JLearningServices *Framework* oferece solução para este problema, na medida em que novas aplicações síncronas poderão ser rapidamente geradas com alto grau de confiabilidade.

4.1 - Introdução

O processo de desenvolvimento de um *Framework* depende do grau de experiência da organização no domínio do problema. Uma organização com maior experiência poderá adotar um processo mais avançado no desenvolvimento [4]. Neste trabalho, optou-se por utilizar um processo baseado na análise do domínio em virtude da experiência já adquirida no domínio do problema.

Este processo caracteriza-se por:

- analisar o domínio do problema para identificar e entender possíveis abstrações a partir do estudo de aplicações existentes,
- identificar-se as abstrações e
- desenvolver-se o *Framework* juntamente com uma aplicação de teste, modificando o *Framework* quando necessário, revisando as aplicações anteriores para verificar se continuam funcionando.

O modelo será visto e comentado em partes de forma a proporcionar um melhor entendimento. Nesta descrição será destacado o *kernel* e os *hot-spots* (bordas em negrito) do ambiente.

4.2 - Descrição do *Framework* JLearningServices

Na **Figura 2** é mostrada a relação entre o servidor (classe *Server*) e as conexões (classe *Connection*). O servidor pode receber e controlar diversas conexões. Foi utilizado o *Design Pattern State* [8] para definir o estado de uma conexão - aberta ou fechada. Esse padrão comportamental permite que o comportamento de um objeto seja modificado de acordo com o estado em que ele se encontra. Isso pode permitir que se remova conexões "mortas", fazendo com que o servidor trabalhe de maneira mais eficiente. Este *Pattern* também foi utilizado para definir os possíveis estados do servidor: em funcionamento, esperando o recebimento de conexões e fechado. Assim tem-se um maior controle sobre o funcionamento do servidor.

O *Design Pattern Observer* [8] foi utilizado para notificar os clientes (classe *Client)* envolvidos em uma sessão de *chat*, por exemplo, a respeito de alterações que ocorram em um algum cliente específico. Esse padrão comportamental permite que, havendo a modificação de um objeto, os demais sejam notificados e atualizados em tempo de execução.

Um padrão também foi utilizado para controle das requisições de *login*, necessárias para que um sistema seja acessado. Esse controle é feito pelo *Design Pattern Chain of Responsibility* [8], um modelo comportamental que permite que várias requisições sejam feitas e acumuladas até que possam ser atendidas. Nesse contexto, existe ainda a classe *UserInformation*, responsável pelo armazenamento dos dados dos usuários, que precisam ser consultados pelo servidor no momento de efetuar o *login*.

Na **Figura 3**, são apresentados os *hot-spots* que podem ser utilizados para todos os tipos de cliente. A classe *Resource* é especializada nas classes *Sound*, *Image*, *Icon*, *ActionButton*, *Text*, *Application* e *Graphic*, permitindo que sejam definidos recursos para uso de sons, imagens, ícones, botões de ação, texto, aplicações e elementos gráficos, respectivamente. Existe também a classe *Action*, associada à classe *ActionButton* para permitir a definição de uma ação para um botão.

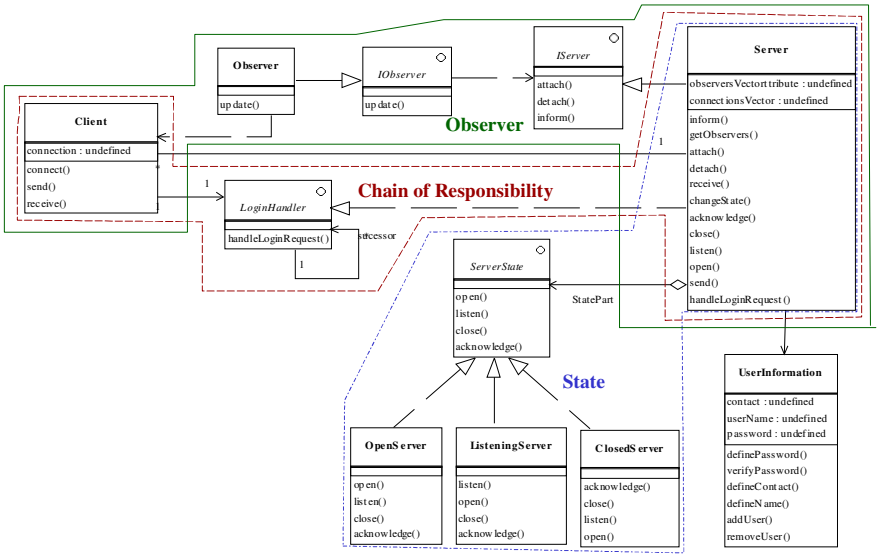

Figura Erro! Indicador não definido. **- Visão parcial do modelo com os *Design Patterns Observer*, *Chain of Responsibility* e *State*.**

Para a definição dos elementos gráficos foi utilizado o *Design Pattern Composite* [8]. Assim, pode ser feito um desenho (classe *Picture)* composto por qualquer combinação de elementos que especializam a classe *Graphic*.

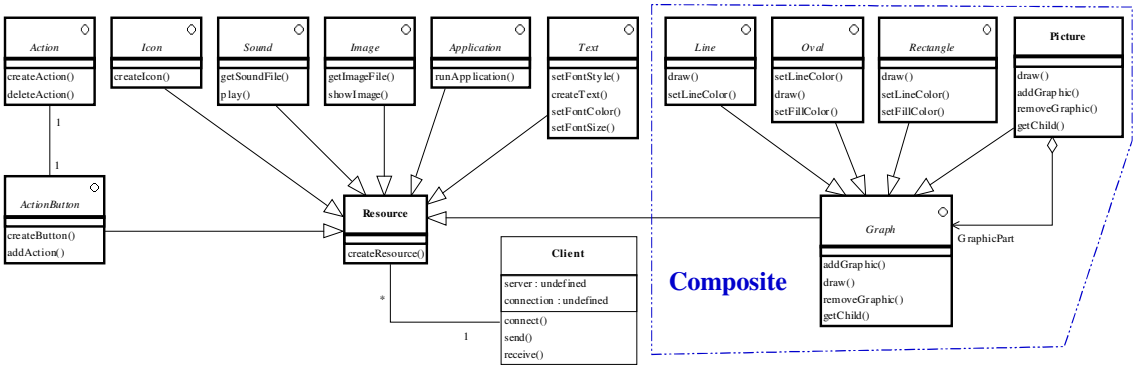

Figura 3 - *Hot-spots* da Classe *Client*.

Os tipos de cliente que podem ser gerados com uso do *Framework* são mostrados na **Figura 4**. A classe *Client* possui especializações de acordo com o tipo de comunicação utilizado pela aplicação que será gerada: um-para-um, um-para-muitos ou muitos-para-muitos. Nesta etapa foi utilizado o *Desing Pattern Application Types* [9], que separa aplicações em categorias de acordo com seu comportamento e características. Aplicações muitos-para-muitos (classe *ClientManyToMany*), semelhantes a *chats*, podem ter uma ou mais salas (classe *Room*) para interação entre diversos participantes (classe *Participant*). As salas podem ser pré-definidas ou criadas por algum participante. O número máximo de participantes em uma sala também pode ser configurado. Uma sala pode ser bloqueada ou liberada para interação. Existe também a possibilidade de manter conversa privada entre participantes, o que é tratado pela classe *PrivateRoom*. Uma sessão de *chat* (ou de qualquer outro tipo de aplicação) pode ser gravada em um arquivo de *log* (classe *SessionLog*) e posteriormente manipulado de acordo com o interesse do usuário - manter o conteúdo parcial/completo do *log* ou deletá-lo. A classe *SessionLog* foi especializada

para contemplar a gravação desses arquivos por qualquer um dos tipos de aplicação (cada tipo de aplicação implementa esse recurso de acordo com suas particularidades).

Aplicações um-para-muitos (classe *ClientOneToMany*) consistem de uma sessão de troca/recebimento de mensagens (classe *ClientOneToManySession*) que é válida enquanto o usuário estiver conectado. Um usuário (classe *User*) recebe de outros solicitação para manter contato, as quais podem ser aceitas ou rejeitadas. Da mesma forma, pode solicitá-la a outros e ser aceito ou rejeitado. Aplicações um-para-um (classe *ClientOneToOne*) também possuem uma sessão de envio/recebimento de mensagens (classe *ClientOneToOneSession*), porém essa interação só pode ocorrer entre duas pessoas (classe *SenderReceiver).*

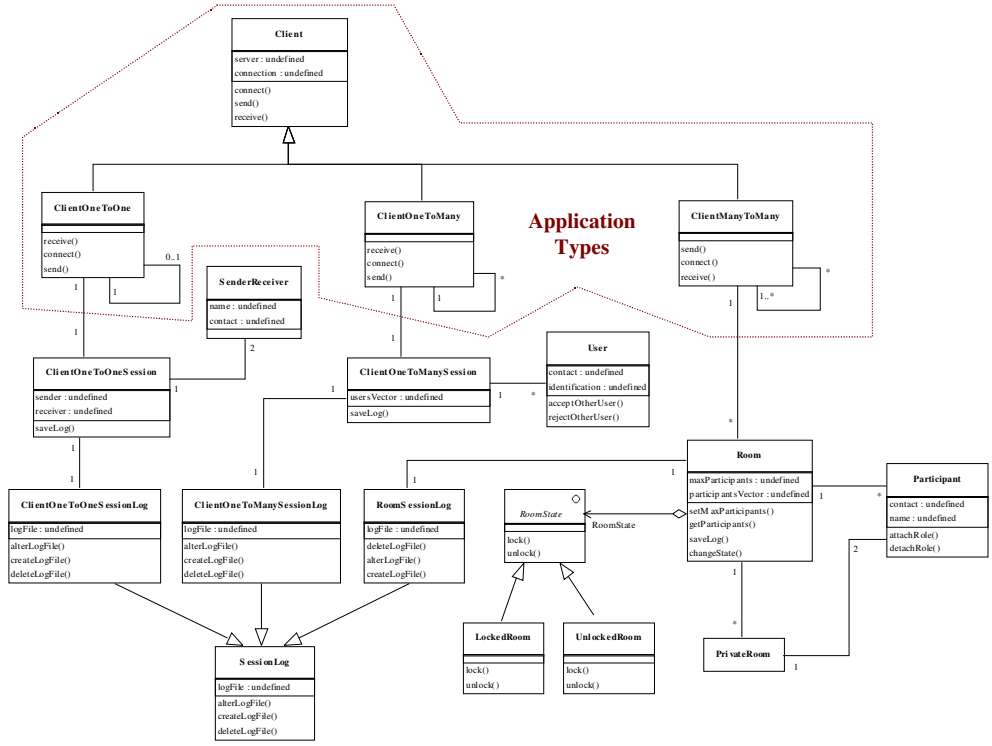

Figura 4 - Flexibilização dos tipos de Clientes modelados com o *Design Pattern Application Types*.

Na **Figura 5**, estão definidos os *hot-spots* referentes a aplicações muitos-para-muitos (classe *ClientManyToMany*). Trata-se dos papéis (classe *Role*) que podem ser assumidos por um participante (classe *Participant*).

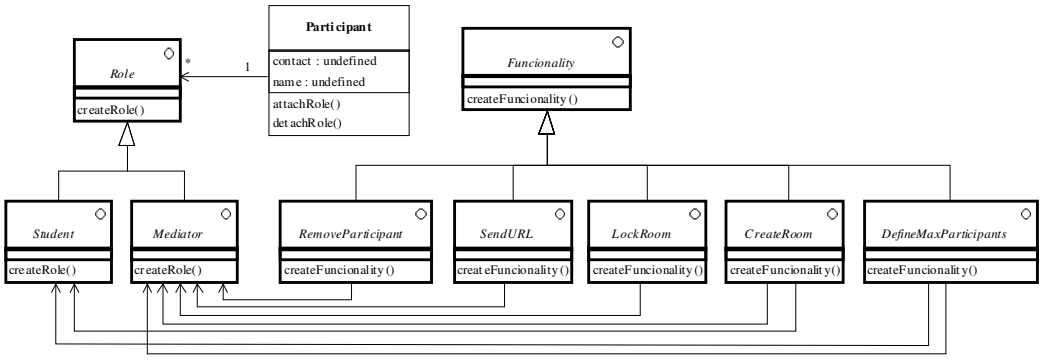

Figura 5 - Visão parcial do modelo com os *hot-spots* referentes às aplicações muitos-para-muitos.

Dois papéis existentes são *Mediator* e *Student*, mas outros podem ser definidos através da especialização da classe *Role* por meio de herança. A cada papel podem estar associadas diversas funcionalidades (classe *Funcionality*) como, por exemplo, o de retirar um participante da sala. Existem funcionalidades definidas – *RemoveParticipant*, *SendURL*, *CreateRoom*, *LockRoom* e

DefineMaxParticipants – mas, assim como a classe *Role*, a classe *Funcionality* pode ser especializada para a adição de novas funcionalidades, as quais podem ser associadas a um ou mais papéis. Todos os *hot-spots* do modelo possuem métodos para sua criação e manipulação, e por serem classes abstratas serão implementados na ocasião da instanciação do *Framework*.

5 - Conclusão e Trabalhos Futuros

A tecnologia de *Frameworks* aliada ao uso *Design Patterns* para desenvolvimento de sistemas proporciona redução de tempo e custo, com aumento da qualidade do *software* produzido. Pode-se obter diferentes aplicações (pertencentes a um mesmo domínio) sem a necessidade de novo trabalho de análise, projeto e codificação de todo o ambiente. Apenas se define o comportamento dos *hot-spots* desejados para aquela aplicação.

Essa união de tecnologias se mostra adequada para áreas onde os requisitos mudam rapidamente, como o Ensino a Distância. Com seu uso, novas funcionalidades podem ser mais facilmente integradas a uma ferramenta do que através do método tradicional de desenvolvimento orientado a objetos.

A metodologia utilizada (processo baseado na análise de domínio) mostrou-se consistente para o desenvolvimento do JLearningServices, uma vez que havia experiência adquirida no domínio de problema ao qual o *Framework* está relacionado.

Em se tratando de ferramentas síncronas, o processo de análise de requisitos não apresentou problemas no que diz respeito à estrutura das aplicações, por se tratarem de tipos de sistemas conhecidos. Porém, praticamente não existem aplicações desse tipo especificamente destinadas ao ensino. Assim, a busca de informações junto a pessoas ligadas à área do ensino, bem como a busca de aplicações que tivessem o maior número de características aproximadas a esse domínio, foi essencial para complementar a visão dos requisitos necessários ao *Framework*.

A principal contribuição do trabalho é o desenvolvimento de um *Framework* para a área de EAD, carente de recursos, e que permitirá a geração de serviços síncronos em seus diferentes tipos de comunicação promovendo flexibilização e reutilização de *software* através do uso de *Frameworks* e *Design Patterns*.

O ambiente está sendo desenvolvido segundo uma metodologia de desenvolvimento em espiral. As fases seguintes são de implementação e instanciação do protótipo, utilizando-se a linguagem Java. Recursos de EAD vêm sendo incorporados ao *Framework* durante seu desenvolvimento.

Este trabalho está vinculado ao projeto WebComposeJ: Uma linguagem para Composição de Serviços Web em Espaços Compartilhados, que tem enfoque em *Web-based education*.

6 - Referências

[1] Paul G. Bassett. “Framing Software Reuse – Lessons From the Real World”. Yourdon Press Computing Series, 1997.

[2] BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. “The Unified Modeling Language user guide”. Massachusetts: Addison-Wesley, 1999.

[3] BRITAIN, Sandy; LIBER, Oleg. “A framework for pedagogical evaluation of virtual learning environments”. University of Wales - Bangor.

[4] Crespo C. S. Pinto, Sérgio. “Composição em WebFrameworks”. Tese de Doutorado, PUC-Rio, Departamento de Informática, 2000.

[5] Crespo, S.; Da Fontoura, M.F.; P. de Lucena, C.J., “Using Viewpoints, frameworks, and domain-specific languages to enhance software reuse”. European Reuse Workshop - ERW'98, Madrid, Spain, 1998.

[6] S. Crespo, M. F. Fontoura, and C. J. Lucena. “A Web-based Educational Environments Comparison using a Conceptual Model compatible with the EDUCOM/IMS Platform”.

Brazilian Symposium on Education and Computer Science (SBIE'98), Fortaleza, Brazil, 1998 (in Portuguese).

[7] S. Crespo, C. J. Lucena, Staa, Arndt V. "FrameConstructor: uma Ferramenta para Suporte a Construção Sistemática de Frameworks". In: Monografias em Ciências da Computação, Depto de Informática PUC-RIO, Rio de Janeiro. MCC42/98, PUC-Rio. 1998.

[8] Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John. "Design patterns: elements of reusable object-oriented software". Massachusetts: Addison-Wesley, 1995.

[9] FAYAD, Mohamed; Schmidt, Douglas; Johnson, Ralph. "Building application frameworks: object-oriented foundations of framework design". New York: John Wiley & Sons, 1999.

[10] Fayad, Mohamed E., Schimidt, C. Douglas and Johnson, E. Raph. "Implementing Application Frameworks – Object-Oriented Frameworks at Work", Chapter 18, Wiley, New York, 1999.

[11] M. F. Fontoura, E. H. Haeusler, and C. J. Lucena. "A Framework Design and Instantiation Method Based on Viewpoints". MCC/98, Monografias em Ciência da Computação, Departamento de Informática, PUC-Rio, August 1998.

[12] Fontoura, Marcus; Crespo, Sérgio; Lucena, Carlos José; Alencar, Paulo; Cowan, Donald. "Using viewpoints to derive object-oriented frameworks: a case study in the web-based education domain". The Journal of Systems and Software, no. 54, Amsterdam, 2000.

[13] Froehlich, Garry; Hoover, James; Liu, Ling; Sorenson, Paul. "Designing Object-Oriented Frameworks". University of Alberta, Canada.

[14] Koskimies, Kai; Mössenböck, Hanspeter. "Designing a Framework by Stepwise Generalization". Proceedings of the 5th European Software Engineering Conference. Lecture Notes in Computer Science 989. Springer-Verlag, 1995. 479-497.

[15] Kreutz, Reinhard; Kiesow, Sven; Spitzer, Klaus. "NetChat: communication and collaboration via WWW". Germany: Technical University Aachen - Institute for Medical Computer Science.

[16] Bent Bruun Kristensen. "Roles: Conceptual abstration theory and practical language issues". Special Issue of theory and Practice of Object Systems (TAPOS) on Subjectivity on Object-Oriented systems, 1996.

[17] Lucena, Carlos; Fuks, Hugo. "Professores e aprendizes na Web: a educação na era da Internet". Rio de Janeiro: Clube do Futuro, 2000. 160 p.

[18] Markiewicz, Marcus E.; Lucena, Carlos. "Issues on Object-Oriented Framework Development". ACM Crossroads, 7.4 , Summer 2001.

[19] Michael Mattsson, "Evolution and Composition Object-Oriented Frameworks". PhD Thesis, University of Karlskrona/Ronneby, Department of Software Engineering and Computer Science, 2000.

[20] W. Pree. "Design Patterns for Object-Oriented Software Development". Addison-Wesley, 1995.

[21] Douglas C. Schmidt, Mohamed Fayad, and Ralf Johnson. "Software Patterns". Communications of the ACM, 39(10), October 1997.

[22] Amjad Umar. "Application (RE) Engineering Building Web-Based Applications and Dealing with Legacies". Prentice Hall PTR, 1997.