6
Especificação

Este capítulo apresenta a especificação do MC2, além da especificação do *framework* Canais de Comunicação. A especificação do *framework* Canais de Comunicação é importante porque o MC2 é uma instanciação deste *framework.*

Um *framework* agiliza o processo de desenvolvimento de *software* que lida com determinado tipo de problema, provendo recursos que fornecem flexibilidade para os desenvolvedores adequarem-no às suas necessidades, reutilizando a solução da parte comum dos problemas. De acordo com [Johnson, 1997] , um *framework* pode ser definido como um projeto reutilizável de todo ou de parte de um sistema, fornecendo um conjunto de classes abstratas e a forma com que suas subclasses interagem. Deste modo, um *framework* pode ser encarado como um esqueleto de uma aplicação que pode ser adaptado por um desenvolvedor. Um *framework* é adaptável a uma situação específica em pontos denominados *hot-spots*, apresentando implementações prototípicas para domínios de conhecimento. [Fuks, Raposo & Gerosa, 2002]

As classes desenvolvidas foram nomeadas em inglês para seguirem as normas de nomenclatura especificadas para o projeto AulaNet.

6.1
Framework Canais de Comunicação

O *framework* Canais de Comunicação [Ferraz 2000] que foi desenvolvido no Laboratório de Engenharia de Software da PUC-Rio em 2000, tem por objetivo prover uma estrutura flexível de sessão e aplicação para sistemas de objetos distribuídos, organizados em espaços (comunidades) virtuais, que se comuniquem através de canais de comunicação.

O *framework* foi dividido em três *framelets* independentes, que podem ser usados sozinhos ou em conjunto para prover uma estrutura de complexidade variável para aplicações que utilizem comunicação de uma maneira geral.

6.1.1 *Framelet* Distribuição

O *framelet* Distribuição tem por objetivo implementar uma interface mais simples do que a encontrada em transporte para o estabelecimento de circuitos virtuais entre objetos. Este *framelet* abstrai os aspectos relativos à comunicação entre cliente e servidor, que podem ser implementadas usando qualquer protocolo de comunicação de transporte.

A Figura 19 mostra o diagrama de classes produzido na fase de projeto do *framelet* Distribuição.

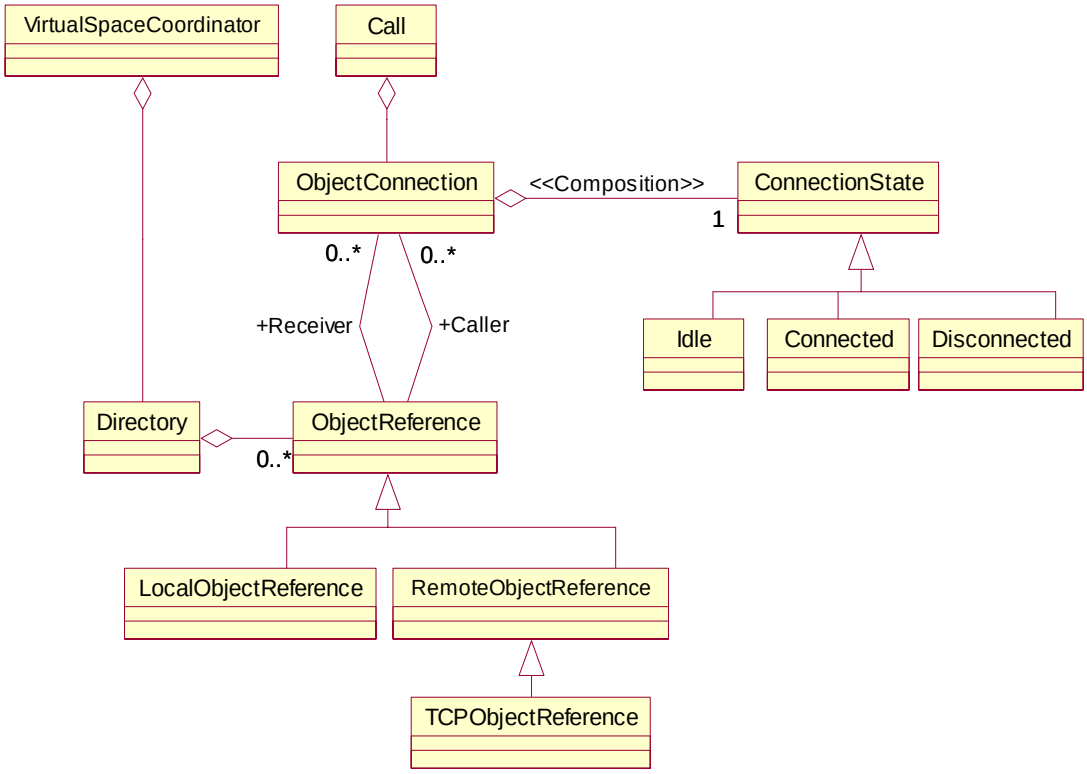

Figura 19 - Diagrama de Classes do Framelet Distribuição

No projeto deste *framelet* foram identificadas as entidades e os papéis a seguir:

Virtual Space Coordinator (Coordenador do Espaço Virtual): Coordena o espaço virtual de objetos, servindo referências virtuais a objetos registrados e provendo serviços de admissão e registro.

Call (Ligação): Implementa com uma *interface* simples, uma ligação em circuito virtual entre dois objetos, fim a fim e sem erros. A ligação implementa o conceito de sessão de comunicação, permitindo que marcadores sejam inseridos ao longo da conversação.

Object Connection (Conexão de Objeto): Garante que a comunicação entre as partes de uma ligação aconteça corretamente, cuidando para que não existam conflitos como os provenientes de concorrência (implementando exclusão mútua) e mantendo o estado da conexão.

Connection State (Estado de Conexão): Implementa o comportamento da conexão de objetos dependendo do estado que esta conexão se encontra – desocupado, conectado ou desconectado.

Directory (Diretório): Implementa as operações de diretório de referências virtuais.

Caller (Chamador): É o papel do objeto que inicia uma conexão (cliente).

Receiver (Receptor): É o papel do objeto que atende um pedido de conexão (servidor).

Object Reference (Referência de Objeto): Implementa a *interface* entre a camada de sessão e a de transporte, lidando com os aspectos específicos de cada protocolo. Referências de objeto podem ser locais ou remotas. Referências locais são para objetos que residem no espaço local do servidor (i.e. coordenador do espaço virtual). Esta classe é disponibilizada para que a comunicação entre objetos da mesma máquina seja possível sem a passagem pela camada de transporte, o que tornaria a comunicação menos eficiente. Referências remotas são

para objetos que residem em outros espaços de memória que não são o servidor. Uma implementação padrão é oferecida para o transporte TCP.

6.1.2
Framelet Canais de Comunicação

O objetivo deste *framelet* é abstrair os aspectos essenciais da comunicação entre publicadores e assinantes através de canais de comunicação, e de prover meios para que um terceiro ator, chamado mediador, possa controlar esta interação. O domínio modelado por este *framelet* é a comunicação. Os conceitos fundamentais envolvidos serão descritos a seguir.

Comunicação pode ser, de maneira geral, dividida em dois grupos, síncrona e assíncrona como mostra a Tabela 1.

	Síncrona	**Assíncrona**
Quadro de Tempo	Tempo real (segundos)	Indefinido (minutos, horas)
Transferência de Conteúdo	Quando disponível	Quando requisitado
Analogia com Meios Tradicionais	Telefonia (bidirecional) e televisão (unidirecional)	Quadro de avisos, secretária eletrônica
Nova Tecnologia	Videoconferência, *shared whiteboard*, *chat*	Correio eletrônico, Real Vídeo

Tabela 1 – Comunicação síncrona e assíncrona

A Figura 20 mostra o diagrama de classes produzido na fase de projeto do *framelet* Canais de Comunicação.

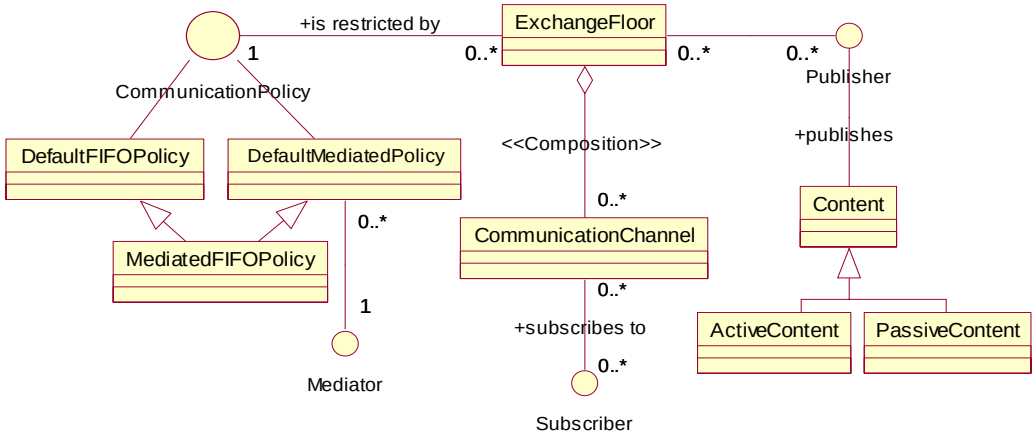

Figura 20 - Diagrama de Classes do Framelet Canais de Comunicação

Existem três atores envolvidos em uma sessão de bate-papo, que são: o publicador, o assinante e o mediador [Ferraz 2000].

O **Publicador** (*Publisher*) é aquele que publica conteúdo em um canal de comunicação. Ele compartilha conteúdo de sua autoria com os demais assinantes de um determinado canal. O publicador depende de autorização do contexto de comunicação para poder publicar para os canais de comunicação deste.

O **Assinante** (*Subscriber*) é quem subscreve para um canal de comunicação, sendo notificado de todo conteúdo publicado neste pelos publicadores. Ele depende de autorização do canal de comunicação para poder subscrever para este.

O **Mediador** (*Mediator*) é o criador de um contexto de comunicação. É ele quem define a política de comunicação do contexto criado por ele. O mediador é capaz de interromper a publicação de conteúdo em um contexto de comunicação, retomando seu controle.

Neste *framelet*, as políticas de comunicação determinam três fatores: quem pode publicar conteúdo nos canais de comunicação, quando isto deve acontecer e quem tem permissão para subscrevê-los. As políticas propostas em [Ferraz 2000] são *DefaultFIFOPolicy*, *DefaultMediatedPolicy* e *MediatedFIFOPolicy*, porém apenas a primeira foi implementada no MC1. As outras duas políticas tiveram suas características implementadas no MC2 através da técnica “Contribuição Mediada”.

Na política *DefaultFIFOPolicy* qualquer pessoa pode se tornar assinante ou publicador dos canais de comunicação e uma fila de publicadores é criada de forma a dar a palavra sempre ao próximo da fila, automaticamente.

Já na *DefaultMediatedPolicy*, a participação do mediador é fundamental. Ele determina quem pode publicar conteúdo nos canais de comunicação, quando isto deverá acontecer e quem terá permissão para lê-los.

A política *MediatedFIFOPolicy* funciona de maneira similar à *DefaultFIFOPolicy*. Entretanto, o mediador pode, a qualquer momento, alterar a ordem de participação padrão. Esta política, embora ainda mediada, exige menos intervenção por parte do mediador.

6.1.3
Framelet Notificação

Este *framelet* provê uma estrutura de suporte à notificação e observação de eventos, seguindo em linhas gerais o padrão de projeto Observador.

Eventos são, em última análise, notificações da alteração do estado de algum objeto. A este objeto dá-se o nome de notificador. De outro lado está o observador, nome dado ao interessado em ser notificado da ocorrência de um determinado tipo de evento com algum objeto notificador em particular.

A Figura 21 mostra o diagrama de classes produzido na fase de projeto do *framelet* Notificação.

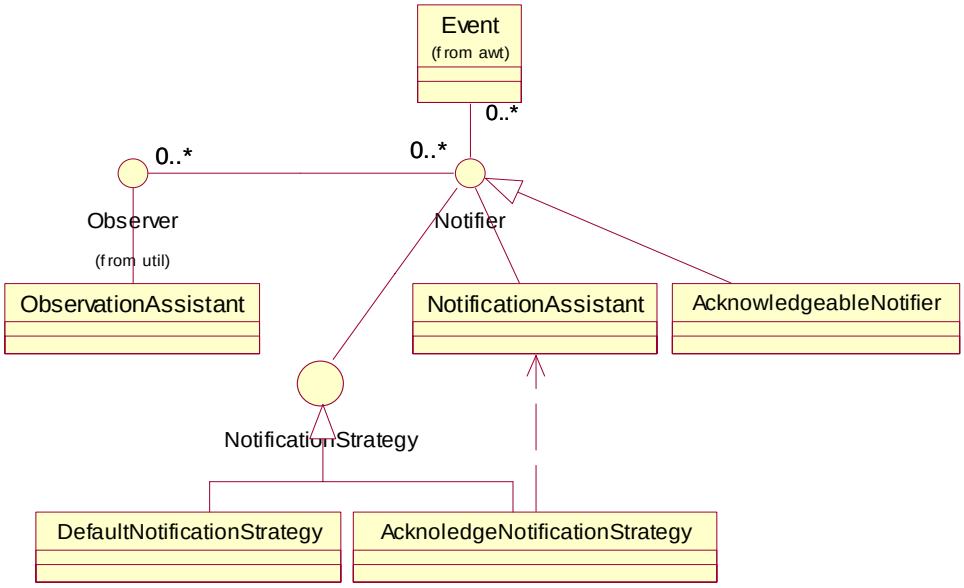

Figura 21 - Diagrama de Classes do Framelet Notificação

Uma das situações mais comuns na comunicação entre notificadores e observadores é a necessidade de se obter a confirmação de que uma determinada notificação (ou mensagem) foi recebida corretamente. Para contemplar este requisito, notificadores podem notificar eventos esperando que seu recebimento seja confirmado. Se for o caso, esta confirmação é enviada para o notificador com a data da entrega com sucesso. Se o aviso de recebimento não for entregue em tempo hábil, especificado previamente pelo notificador como expiração do evento,

este deverá ser avisado do fato de forma que possa tomar as providências necessárias. Eventos expirados podem ser ignorados pelos observadores.

A implementação deste mecanismo esbarra em um problema notório que é o da sincronização de relógios. Para que faça sentido se falar em eventos expirados após uma certa marca no tempo, tanto notificador quanto observador devem possuir relógios sincronizados.

Para que não seja necessário que notificadores e observadores lidem com a complexidade descrita acima, são introduzidos os assistentes de observação e notificação. Caso os observadores e notificadores concretos não precisem especializar o comportamento implementado pelos assistentes, podem reutilizá-los, via agregação, diretamente.

O primeiro *framelet* descrito implementa a camada de sessão descrita no modelo de referência OSI. Os dois seguintes implementam a camada de aplicação, definindo o protocolo sob o qual se dá a comunicação entre objetos.

O *framework* Canais de Comunicação é a junção dos três *framelets* Distribuição, Canais de Comunicação e Notificação que acabaram de ser descritos. Eles podem ser usados em conjunto ou em separado, conforme a necessidade da aplicação.

A Figura 22 apresenta o diagrama de classes do *Framework* Canais de Comunicação que foi desenvolvido por [Ferraz 2000].

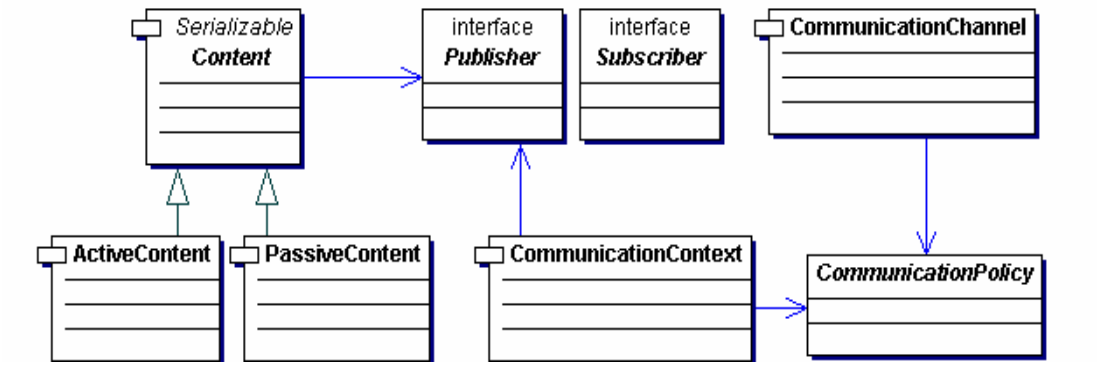

Figura 22 - Diagrama de classes do Framework Canais de Comunicação

Sumário do Diagrama de Classes do *Framework* Canais de Comunicação:

Publisher (Publicador): As classes que implementam esta *interface* são capazes de publicar conteúdo (*Content*) nos canais de comunicação (*CommunicationChannel*). Elas serão notificadas sempre que estiverem capacitadas a publicar para os canais de um contexto de comunicação (*CommunicationContext*). As classes devem notificar ao contexto quando tiverem terminado. Elas serão notificadas quando tiverem que parar de publicar ao contexto.

Subscriber (Assinante): As classes que implementam esta *interface* são capazes de subscrever para os canais de comunicação. Elas serão notificadas sempre que algum conteúdo for publicado no canal por um publicador (*Publisher*).

ActiveContent (Conteúdo Ativo): Conteúdos deste tipo são executáveis.

CommunicationChannel (Canal de Comunicação): É a mídia atual por onde todo conteúdo passa. Notifica os assinantes (*Subscribers*) do conteúdo publicado.

CommunicationContext (Contexto de Comunicação): Agrupa os canais de comunicação, fazendo com que o grupo mantenha a mesma política e obrigando que a concessão dos canais de comunicação seja dada a um único publicador por vez.

CommunicationPolicy (Políticas de Comunicação): Controla a política de comunicação do contexto de comunicação. Decide se alguém pode ser um publicador ou um assinante do contexto e como o próximo publicador é selecionado.

Content (Conteúdo): O que é publicado para os canais. Conteúdos podem ser ativos ou passivos.

PassiveContent**:** Conteúdo passivo.

Esta foi a especificação do *Framework* Canais de Comunicação. É importante frisar que tanto o MC1 como o MC2 são instanciações deste *framework*.

6.2
Especificação do Mediated Chat 2.0

Assim como o MC1, o MC2 foi implementado para a plataforma *Web*. Utilizou-se as linguagens *Java*, HTML e algumas funções em *Java Script*.

A linguagem *Java* foi escolhida, principalmente devido à sua característica de portabilidade e adequação para o desenvolvimento de aplicações para a *Internet*. A simplicidade da linguagem aliada a uma série de recursos disponíveis através de bibliotecas e os mecanismos de segurança principalmente para os programas do tipo *applet* também contribuíram para esta escolha [Rezende & Drummond, 2000]

A combinação da linguagem *Java* com recursos para a programação para a *Internet* oferece as seguintes vantagens:

- Distribuir aplicações e não apenas informações;
- Eliminar a necessidade de portar aplicações;
- Eliminar a necessidade de instalação por parte do usuário final;
- Cortar gastos com a distribuição de *software*;
- Possibilitar a execução de programas localmente;
- Utilizar a rede como veículo de distribuição de aplicações; e
- Facilitar a contínua atualização de aplicações.

Applets são pequenas aplicações que são executadas por um servidor de *Internet*, podem ser transportadas através da rede, instaladas automaticamente e executar *in situ*, como parte de um documento *Web*. Quando um *applet* é instalado em uma máquina cliente, ele possui acesso limitado aos recursos dessa máquina, podendo implementar uma *interface* multimídia arbitrária ou efetuar computações complexas, sem entretanto introduzir risco de contaminação por vírus ou de corrupção da integridade dos dados.

O *applet* é um programa que pode reagir à entrada do usuário e mudar dinamicamente – não apenas executar a mesma animação ou som repetidamente. Como o *applet* é executado no ambiente de um navegador da *Web*, ele não é iniciado diretamente através de um comando. Deve ser criado um arquivo que informe ao navegador o que ele deve carregar e como executar e em seguida, deve-se apontar o navegador no URL que especifica esse arquivo HTML.

Como os *applets* são fragmentos de código carregados na *Web*, eles representariam um perigo inerente. A maneira encontrada por *Java* para evitar que um arquivo seja violado foi através de um modelo de segurança que controla o acesso a praticamente todas as chamadas em nível de sistema na Máquina Virtual *Java* (MVJ). O nível de segurança a ser controlado é implementado no navegador e impede qualquer I/O (entrada/saída) de arquivo, chamadas de qualquer método nativo, e tentativas de abrir um soquete para qualquer sistema, com exceção do *host* que forneceu o *applet*.

Devido as características apresentadas, assim como o MC1, decidiu-se que o MC2 também seria um *applet*. Como foi dito anteriormente, foi preciso criar uma página HTML para carregar e executar o *applet*. É através desta página que são passados alguns parâmetros importantes ao *applet*, que são: nome do participante, identificador do participante, tipo de participante (mediador ou aprendiz), *e-mail* do participante, nome e localização do servidor de bate-papo, tamanho da janela, entre outros. As funções implementadas em *Java Script* são aquelas que estão na barra funcional do Debate: Salvar, Armazena Participantes e Registra Debate.

A seguir serão apresentados os documentos gerados durante as fases de requisitos, análise, projeto e implementação que fazem parte da modelagem do MC2.

6.2.1
Descrição do Problema (MC2)

A descrição a seguir serve apenas para apresentar alguns conceitos presentes na aplicação proposta, não sendo, portanto, completa.

O coordenador de um curso do AulaNet atua como projetista do curso, definindo como será o ambiente virtual onde as atividades do curso se desenrolarão. Ele é o responsável por adicionar e remover uma sala de bate-papo de um curso do ambiente, o que é feito com a inserção ou retirada do serviço Debate, na área de atualização dos Mecanismos de Comunicação, do curso.

Uma sala de bate-papo é um contexto de comunicação onde existem canais de comunicação e membros que usam estes canais. Somente os membros de uma determinada sala podem usar os canais de comunicação desta sala. Este uso é controlado pela política de uso do contexto de comunicação, determinada pelo mediador do grupo.

Canais de comunicação servem para o envio e recebimento de conteúdo (informação). Sobre a ordem das ações dos membros sobre os canais é preciso destacar que apenas um membro pode publicar conteúdo para canais de um determinado contexto de comunicação por vez. Existe, portanto, uma ordem para que os membros tenham direito de enviar conteúdo pelo canal, determinada pela política de uso de contexto de comunicação ao qual o canal pertence. Os membros de uma sala podem ter dois tipos de ações sobre os canais que são enviar e receber conteúdo. É preciso definir quais membros podem enviar e quais podem receber conteúdo pelo canal.

6.2.2
Dicionário de Dados do MC2

A seguir será apresentado o Dicionário de Dados do MC2, que relata a descrição das entidades externas.

Técnica de Conversação Contribuição Livre

- Noção: Determina uma regra que rege a conversação em salas de bate-papo.
- Impacto: Determina que todos os aprendizes podem subscrever para a *sala de bate-papo* e que têm o direito de publicar *conteúdo* nos *canais de comunicação* da sala de bate-papo.

Técnica de Conversação Contribuição Circular

- Noção: Determina uma regra que rege a conversação em salas de bate-papo.
- Impacto: Determina que todos os aprendizes podem subscrever para a *sala de bate-papo* e que cada um na sua vez deve publicar *conteúdo* nos *canais de comunicação* da sala de bate-papo. A ordem de participação é determinada pela aplicação.

Técnica de Conversação Contribuição Única

- Noção: Determina uma regra que rege a conversação em salas de bate-papo.
- Impacto: Determina que todos os aprendizes podem subscrever para a *sala de bate-papo* e que cada um deve publicar *conteúdo* uma única vez nos *canais de comunicação* da sala de bate-papo.

Técnica de Conversação Contribuição Mediada

- Noção: Determina uma regra que rege a conversação em salas de bate-papo.
- Impacto: Determina que todos os aprendizes podem subscrever para a *sala de bate-papo* e que o direito de publicar *conteúdo* nos *canais de comunicação* da sala de bate-papo depende da decisão do mediador.

Sala de Bate-Papo

- Noção: É um tipo de *contexto de comunicação*.
- Impacto: Permite que os *participantes* cooperem e troquem *conteúdo* (informação). Uma sala de bate-papo é criada pelo coordenador. Ela só pode ser uma *sala simples*, ou seja, não possui salas de bate-papo subordinadas.

Participante

- Noção: Membro de uma *sala de bate-papo*, podendo tanto publicar conteúdo como subscrever para os *canais de comunicação* destas. Eventualmente pode assumir o papel de *mediador* da *sala de bate-papo*.

- Impacto: Publica *conteúdo* nos *canais de comunicação* das *salas de bate-papo* ou recebe *conteúdo* dos *canais de comunicação* dos quais é *assinante*.

Sala Simples

- Noção: É um tipo de *sala de bate-papo* que não pode ter subsalas.
- Impacto: *Participantes* não podem criar outras *salas de bate-papo* dentro desta.

6.2.3 Diagramas de Caso de Uso do MC2

Na Figura 23 estão representados os atores do ambiente AulaNet que utilizam o MC2. O coordenador (*Coordinator*) é quem irá adicionar ou remover uma sala de bate-papo de um curso do ambiente. O participante (*Attendee*) que irá usar o MC2 poderá ser um mediador (*Mediator*) ou um aprendiz (*Learner*).

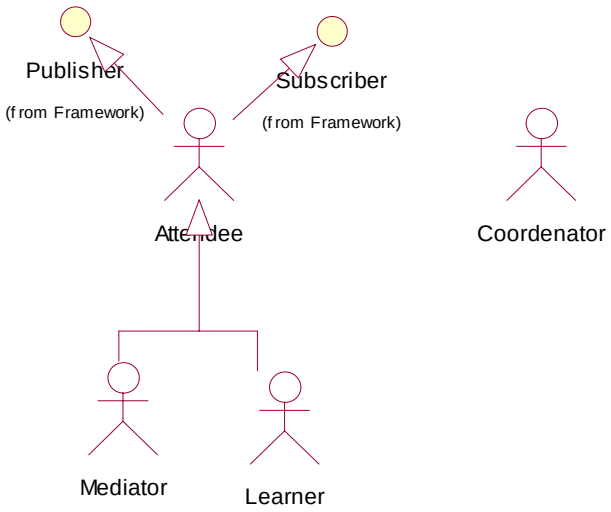

Figura 23 - Atores do AulaNet no MC2

O mediador e o aprendiz possuem *interfaces* distintas, por serem distintos os seus objetivos. O aprendiz utiliza o MC2 para se comunicar com os outros participantes que também estejam usando o MC2. O mediador é o participante que

possui como responsabilidade coordenar a conversação, ou seja, organizar a interação, além de ter que manter um nível de organização da mesma.

A seguir na Figura 24 é apresentado o caso de uso que especifica a utilização do MC2 por seus atores, com o intuito de mostrar a utilização da aplicação.

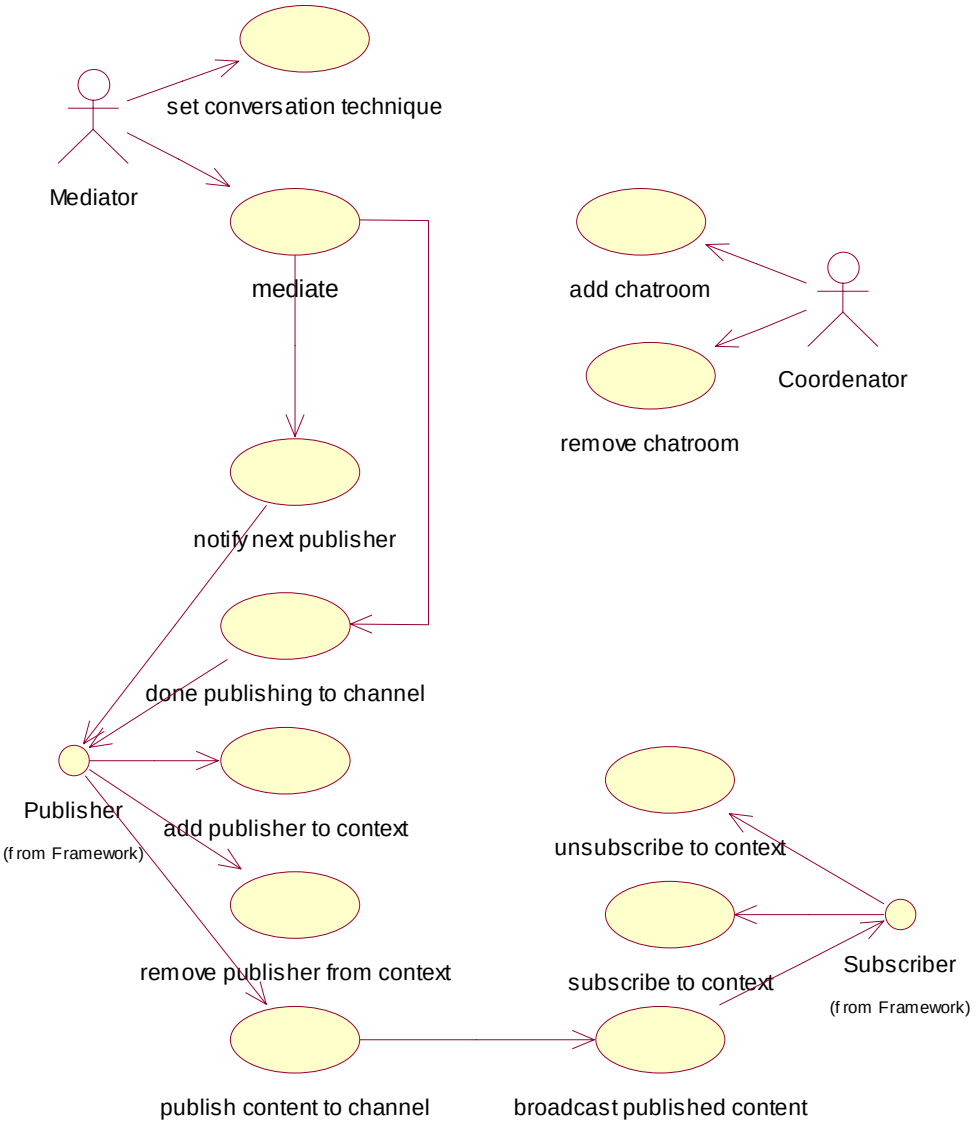

Figura 24 - Diagrama de Caso de Uso do MC2

Como já foi dito anteriormente, o coordenador poderá inserir ou retirar o serviço Debate do curso.

No MC2 o mediador – ator do AulaNet – assume 3 papéis: *mediator*, *publisher* e *subscriber*. Já o aprendiz assume os papéis de *publisher* e *subscriber*. Ao iniciar o uso da aplicação, tanto o mediador quanto o aprendiz adicionam-se como *publisher* e *subscriber* do contexto de comunicação. Eles se removem do mesmo contexto quando saem da aplicação.

O *mediator* tem como função a mediação da conversação entre os aprendizes. Para isso ele pode fazer uso das técnicas de conversação que estão disponíveis na aplicação. Ele também pode mediar a conversação decidindo quem será o próximo aprendiz a publicar no canal de comunicação e quando esta publicação deve ser encerrada.

O *publisher* é quem pode publicar no canal de comunicação. Quando o uso do canal está liberado ele pode publicar quando e se desejar, porém, quando o uso do canal é limitado, ele deve aguardar a sua vez de publicar.

O *subscriber* é aquele que subscreve para o canal de comunicação. No MC2 ele é notificado de todo conteúdo publicado pelos publicadores, ou seja, ele recebe todas as contribuições enviadas pelos participantes.

Na Figura 25 é apresentada uma expansão do caso de uso *setConversationTechnique* que especifica a alteração das técnicas de conversação por parte do mediador.

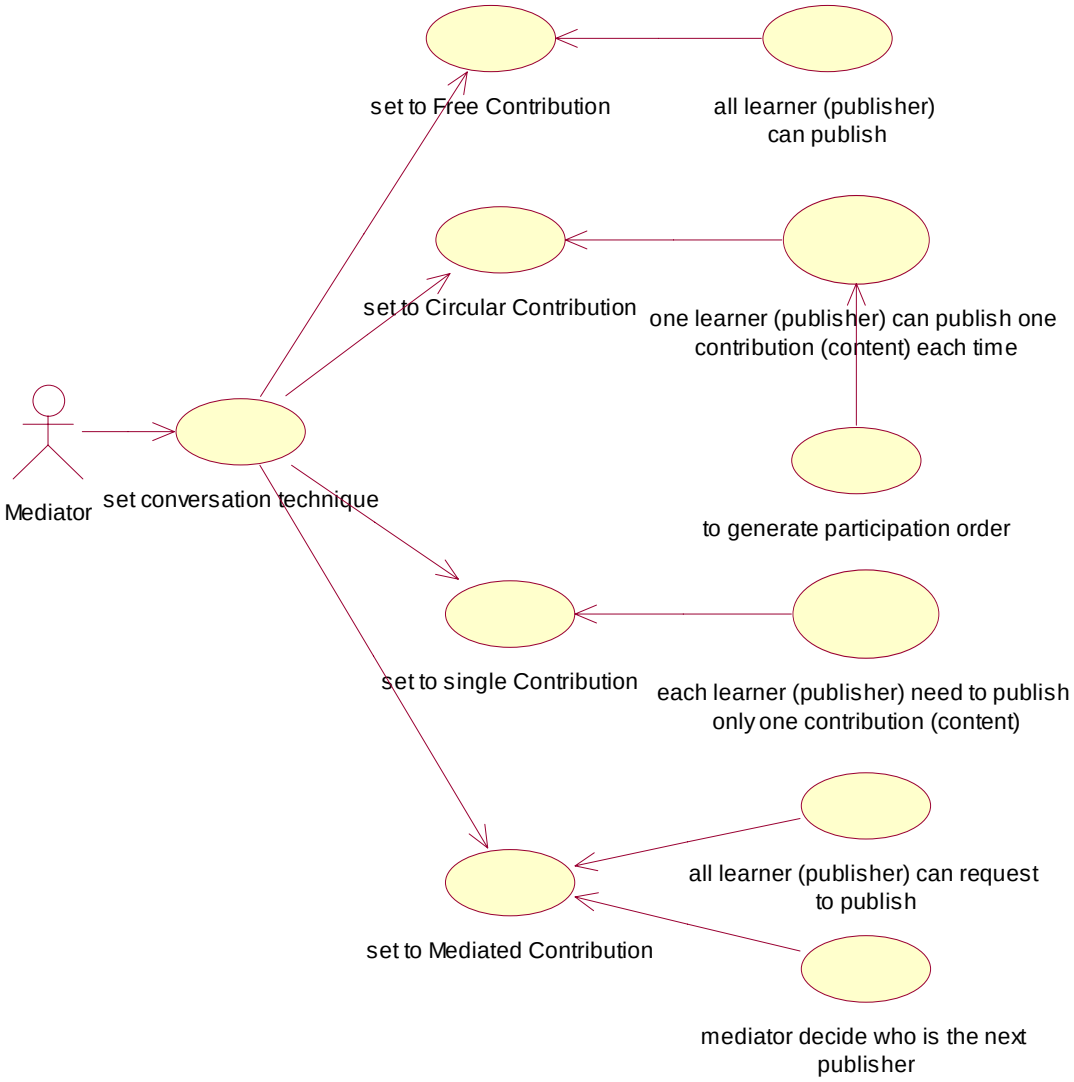

Figura 25 - Expansão do Caso de Uso setConversationTechnique

No MC2 o mediador possui 4 técnicas de conversação a sua disposição. Ao selecionar a Contribuição Livre o mediador deixa o canal de comunicação liberado para que qualquer aprendiz (*publisher*) publique quando e se desejar. Quando o mediador seleciona a Contribuição Circular é gerada, pela aplicação, a ordem de participação, que é a ordem em que o aprendiz deve publicar. Cada aprendiz deve enviar (publicar) uma contribuição (conteúdo) por vez. Isto será controlado pela aplicação pois o MC2 apenas habilita o botão de Enviar de um aprendiz após a publicação de um conteúdo pelo aprendiz que está imediatamente à sua frente na fila da ordem de participação. Após a publicação do conteúdo o aprendiz terá seu botão de Enviar desabilitado, pela aplicação, até que sua vez de publicar chegue novamente. Quando o mediador seleciona a Contribuição Única,

a aplicação limitará em um o número de conteúdos que devem ser publicados por cada aprendiz. Quando o mediador seleciona a Contribuição Mediada, qualquer aprendiz pode solicitar ao mediador o uso do canal de comunicação. Cabe ao mediador decidir quem será o próximo a publicar no canal.

6.2.4
Diagrama de Classes do MC2

Como pode ser visto na Figura 26, o MC2 possui três pacotes que são o ***model***, o ***net*** e o ***view***. O pacote *model* compreende as classes do modelo de negócios do MC2, que são *Attendee, ChatRoom, SimpleRoom, FreeContribution, CircularContribution, SingleContribution, MediatedContribution.* O pacote *net* compreende as classes do modelo de negócios do MC2 responsável pela interface entre o modelo no servidor e no cliente, que são: *ChatRoomServer, Server, Connection, Command, PublishContentCommand, AddAttendeeCommand, RemoveAttendeeCommand, BlockAttendeeCommand,UnblockAttendeeCommand, SetConversationTechniqueCommand.* O pacote *view* compreende as classes da *interface* entre o MC2 e o participante (mediador ou aprendiz). *AttendeeView* é o *applet* que constitui a própria *interface* do participante.

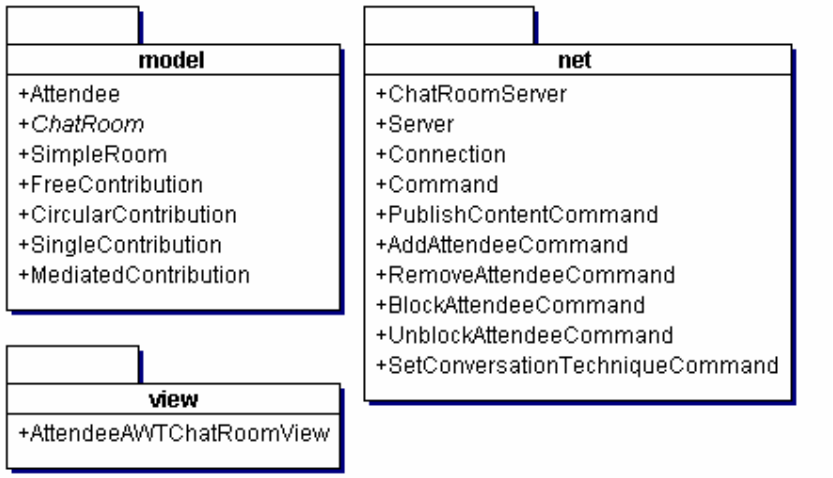

Figura 26 - Pacotes do MC2

A Figura 27 apresenta o diagrama de classes do *Mediated Chat* 2.0 que foi a aplicação desenvolvida para ajudar o mediador na condução da conversação entre os aprendizes de um curso do ambiente AulaNet.

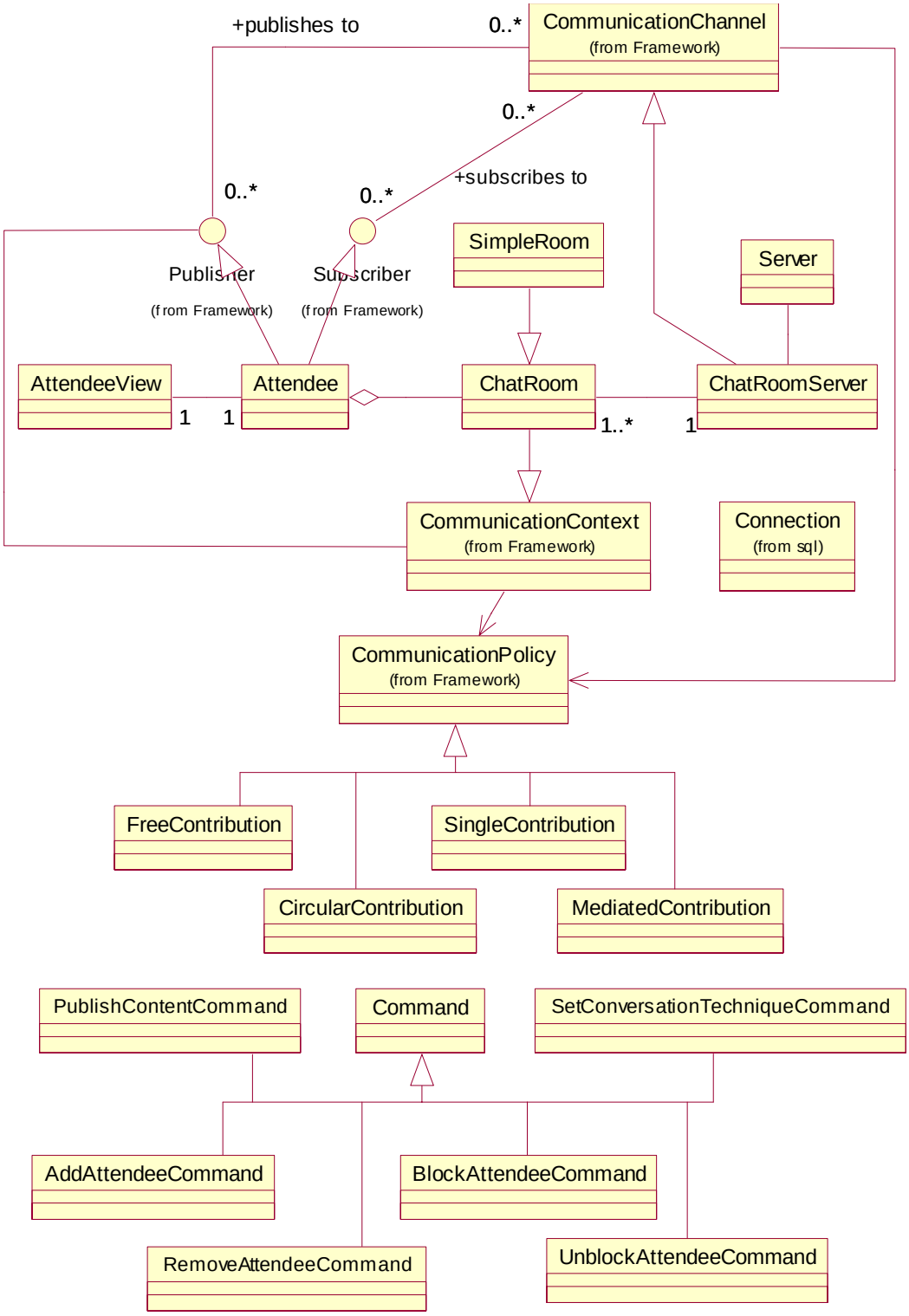

Figura 27 - Diagrama de Classes do MC2

Sumário do Diagrama de Classes do MC2:

Attendee (Participante): Representa um participante da sala de bate-papo. Pode ser um mediador ou um aprendiz.

AttendeeView (Visão do Participante): É a visão (*interface*) do participante na sala de bate-papo. O mediador e o aprendiz possuem *interfaces* distintas.

ChatRoom (Sala de Bate-papo): Extende a classe *CommunicationContext* implementando o *design pattern Composite*.

SimpleRoom (Sala de Bate-papo Simples): É uma sala de bate-papo que não pode ter subsalas.

ChatRoomServer (Servidor de Bate-papo): Extende a classe *CommunicationChannel.*

Server (Servidor): Serviço do Mediated Chat 2.0.

Connection (Conexão): Responsável pela conexão de cada participante.

FreeContribution (Contribuição Livre): Determina que qualquer aprendiz pode publicar para o contexto a qualquer momento.

CircularContribution (Contribuição Circular): Determina que cada aprendiz deverá publicar para o contexto no momento determinado pela aplicação. A participação seguirá a ordem de entrada do participante e se dará de forma circular.

SingleContribution (Contribuição Única): Determina que cada aprendiz deverá publicar para o contexto uma única vez.

MediatedContribution (Contribuição Mediada): Determina que o mediador decidirá sobre a participação ou não de cada aprendiz.

Command (Comando): Comando abstrato que pode ser executado.

PublishContentCommand (Comando Publica Conteúdo): Publica o conteúdo no canal de bate-papo.

AddAttendeeCommand (Comando Adiciona Participante): Pede ao *ChatRoomServer* que adicione o participante à sala de bate-papo.

RemoveAttendeeCommand (Comando Remove Participante): Pede ao *ChatRoomServer* que remova o participante da sala de bate-papo.

BlockAttendeeCommand (Comando Bloqueia Participante): Bloqueia o aprendiz, impedindo-o de publicar. Pode ser extendido a todos os aprendizes, ou seja, bloqueia todos os aprendizes. Este comando só pode ser acionado pelo mediador.

UnblockAttendeeCommand (Comando Desbloqueia Participante): Desbloqueia o aprendiz, permitindo que volte a publicar. Pode ser extendido a todos os aprendizes, ou seja, desbloqueia todos os aprendizes. Este comando só pode ser acionado pelo mediador.

SetConversationTechniqueCommand (Comando Seleciona Técnica de Conversação): Realiza a troca de técnica de conversação. Este comando só pode ser acionado pelo mediador.

É importante ressaltar que o MC2 foi um projeto desenvolvido totalmente independente do ambiente AulaNet, ou seja, com simples modificações ele pode ser usado em qualquer ambiente fora do AulaNet.