4
Reuso em ASCs usando ESSMA: o Framework CAAF

Neste capítulo, apresentamos a principal contribuição de nosso trabalho. O *framework* CAAF é proposto para: (1) facilitar o desenvolvimento de das ASCs em conjunto com outras ferramentas existentes, isto é, com os *middlewares* para ASCs e as plataformas de agentes; (2) aumentar o reuso em ASCs, integrando soluções como MoCA e JADE. CAAF foi abstraído a partir de nossa experiência de reengenharia das ASCs VL e WMS (Seção 2.2) e utilizado para o desenvolvimento de *Health Care* (Seção 4.3).

4.1.
Estrutura de CAAF

As Figuras 21 e 22 apresentam as classes do *framework* CAAF para a instanciação das aplicações servidora e cliente em uma arquitetura que dá suporte à sensibilidade ao contexo. O reuso das funcionalidades de sensibilidade ao contexto e de agentes de software é obtido, porque CAAF integra as infra-estruturadas oferecidas por MoCA e JADE, ao mesmo tempo em que oferece uma solução padrão para o desenvolvimento de ASCs.

CAAF é composto de: (1) classes que representam agentes de usuário no cliente (Figura 22), além de agentes de contexto e de servidor (Figura 21); (2) atributos e métodos que representam elementos do conhecimento interno destes agentes e que são manipuladas por propriedades de agência suportadas através das APIs de JADE; (3) classes que suportam a sensibilidade ao contexto obtidas a partir das APIs de MoCA.

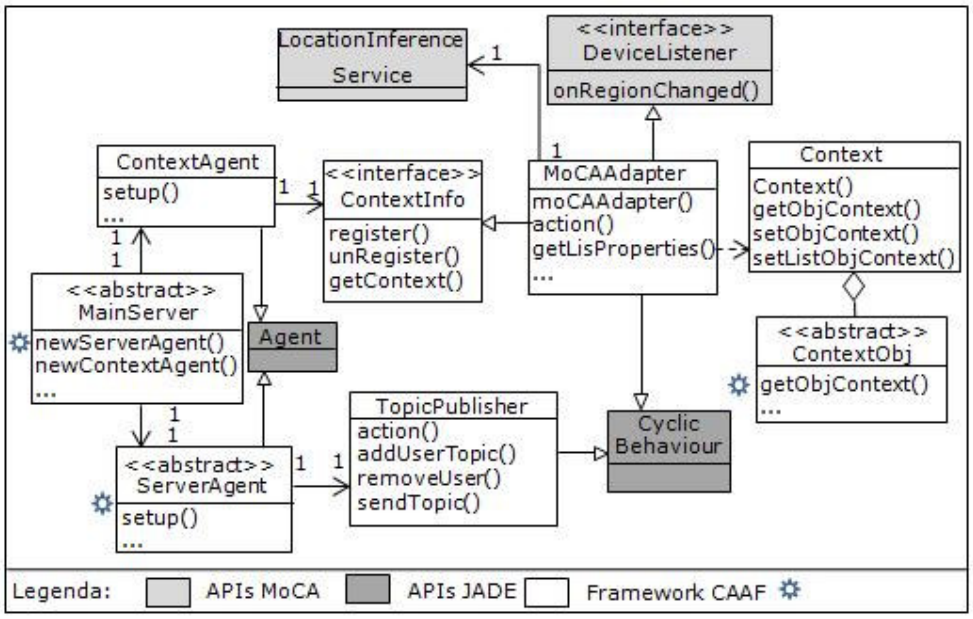

Figura 21. Classes do Framework CAAF para o Servidor da Instância

Na Figura 21, os pontos de flexibilização são as classes `MainServer`, `ServerAgent` e `ContextObj`, utilizadas, respectivamente, para a inicialização da aplicação servidora, a execução de serviços da aplicação e a manipulação abstrata de contextos por tais serviços. Em outras palavras, para a instanciação da aplicação servidora, o usuário de CAAF deve efetuar os seguintes passos (não necessariamente nesta ordem):

1. especificar como subclasses de `ContextObj` os tipos de contexto da aplicação que serão tratados no servidor;
2. especificar serviços da aplicação representados, por exemplo, por uma subclasse de `ServerAgent` e possíveis comportamentos associados;
3. especificar a inicialização de `ServerAgent` na classe `MainServer` e outros possíveis procedimentos de inicialização da aplicação.

Note que são *vários* os pontos fixos ou de reuso em CAAF, os quais constituem a própria estrutura de uma aplicação servidora em ASCs. Por exemplo, as classes `ServerAgent` e `TopicPublisher` são responsáveis pela publicação de tópicos de interesse subscritos pelo cliente. As classes `ContextAgent` e `ContextInfo` são responsáveis pelo interfaceamento da aplicação cliente com o *middleware* que dá suporte à sensibilidade ao contexto (neste caso, MoCA, representado pela classe `MoCAAdapter`).

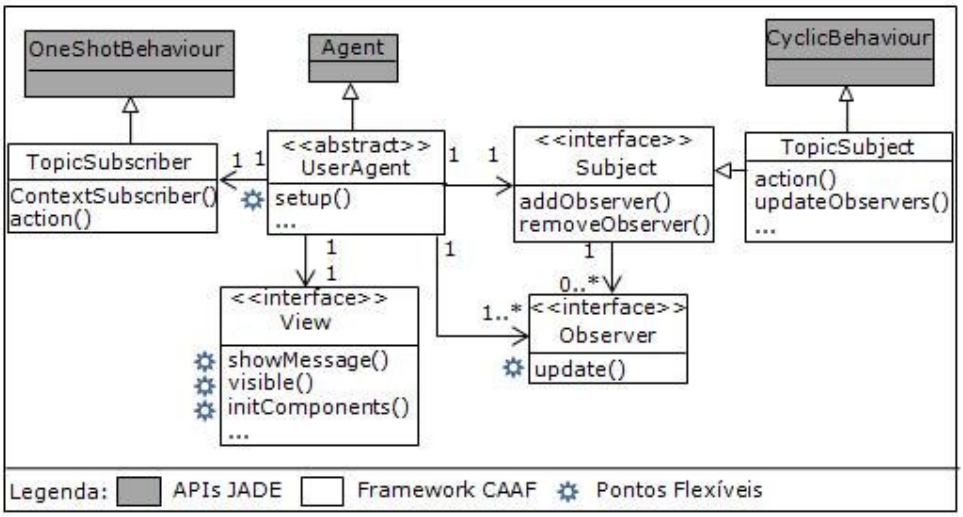

Figura 22. Classes do Framework CAAF para o Cliente da Instância

Na Figura 22, os pontos de flexibilização de CAAF estão presentes na classe `UserAgent` e nas interfaces `View` e `Observer`. De fato, para a instanciação da aplicação cliente, o usuário de CAAF deve efetuar os seguintes passos (não necessariamente nesta ordem):

1. especificar os métodos de `UserAgent`, os quais são basicamente os mesmos métodos abstratos herdados da classe `Agent` (Seção 3.1.4);
2. especificar os métodos de `View`, responsáveis pela implementação da interface gráfica da aplicação cliente;
3. especificar o tratamento dado a tópicos de interesse do cliente no método abstrato `update()` da interface `Observer`.

Note que os pontos fixos de CAAF consistem na própria estrutura oferecida para a instanciação da aplicação cliente em ASCs, bem como na subscrição flexível de seus tópicos de interesse na aplicação servidora (Figura 21). Quanto à estrutura, por exemplo, a interface gráfica é gerenciada abstratamente pelo agente de usuário através da manipulação da interface `View` pela classe `UserAgent`. Já quanto à subscrição de interesses, o agente de usuário especifica tal comportamento nas classes `TopicSubscriber`, `TopicSubject` e `TopicObserver`.

4.2.
Exemplo de Instanciação de CAAF

A aplicação *HealthCare* (HC) oferece suporte ao monitoramento de pacientes com doença cardiovascular sob cuidados médicos em UTIs. Os equipamentos que monitoram funções vitais, como pressão arterial e batimentos por minutos, e analisam tais funções a fim de encontrar quaisquer anomalias no quadro clínico de pacientes (aplicação servidora). Caso algo anormal seja encontrado, o equipamento é responsável por enviar um alerta ao médico (aplicação cliente), que também pode consultar outras informações a respeito do quadro dos pacientes. As Figuras 23 e 24 apresentam os diagramas de classes de HC obtidos a partir da instanciação de CAAF.

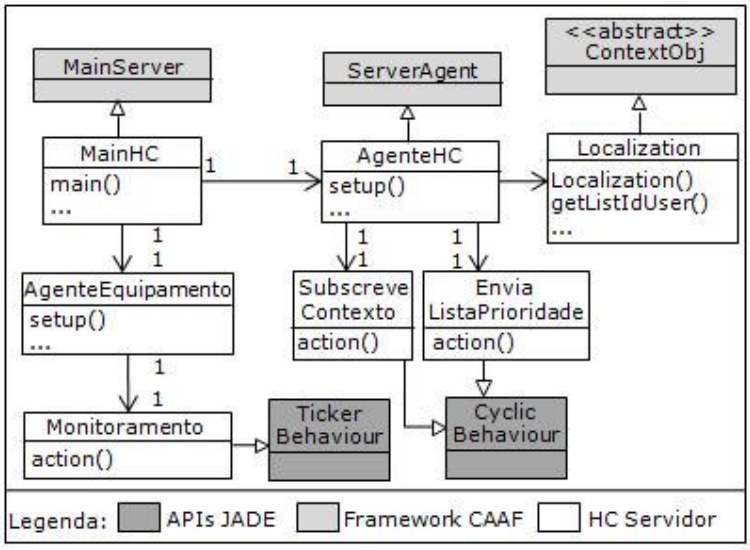

Figura 23. Diagrama de Classes da Aplicação Servidora de *HealthCare*

Na Figura 23, estão presentes apenas as classes de CAAF que possuem pontos de flexibilização, além das classes da aplicação servidora de HC. Em outras palavras, para chegarmos à instância do servidor de HC, especificamos:

1. o tipo de informação de contexto manipulada em HC, neste caso, a classe `Localization` é especificada como subclasse de `ContextObj`;
2. o tipo de `ServerAgent` em HC, isto é, a classe `AgenteHC` e seus comportamentos associados, como `EnviaListaPrioridade`;
3. o tipo `MainHC`, onde é especificada a inicialização de um `AgenteEquipamento`, responsável pela simulação da observação das funções vitais de um paciente.

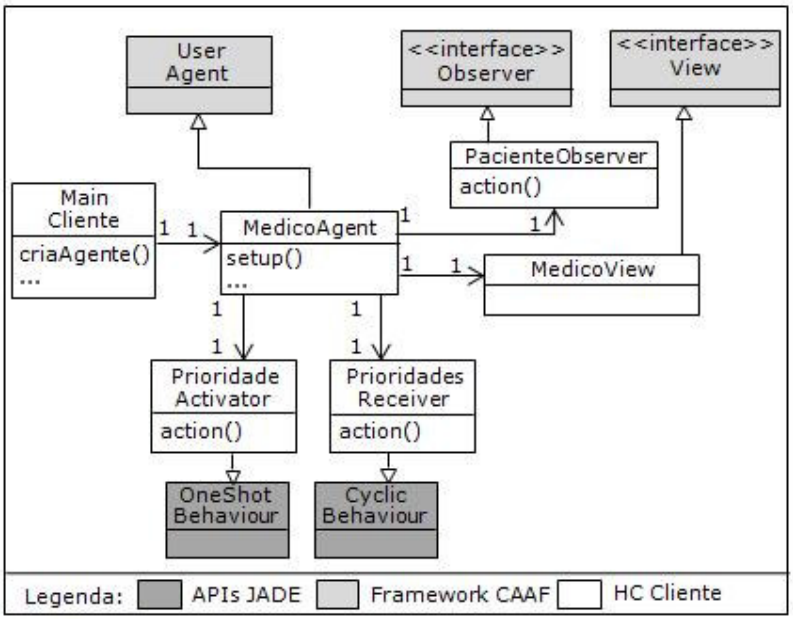

Figura 24. Diagrama de Classes da Aplicação Cliente de *HealthCare*

Por sua vez, na Figura 24, estão presentes apenas as classes de CAAF que possuem pontos de flexibilização, além das classes específicas da aplicação cliente de HC. Para instanciar o cliente de HC, especificamos:

1. o agente de usuário da aplicação, neste caso, a classe `MedicoAgent` é especificada como subclasse de `UserAgent`, além dos comportamentos do agente, como a classe `PrioriedadeActivator`;
2. a interface gráfica da aplicação, isto é, a classe `MedicoView`;
3. o observador da aplicação, isto é a classe `PacienteObserver`.

Das Figuras 23 e 24, vemos que, internamente, *Health Care* se constitui basicamente pela interação entre os agentes de servidor e de usuário da aplicação. Um agente de servidor representa o equipamento responsável pela medição de informações do paciente. Este agente também é responsável por publicar aos interessados as informações de contexto coletadas. Já o agente de usuário representa um médico que tem interesse nas informações médicas dos pacientes. A Figura 25 ilustra o envio ao médico da lista de pacientes com prioridade de atendimento de acordo com as informações de contexto especificadas por ele.

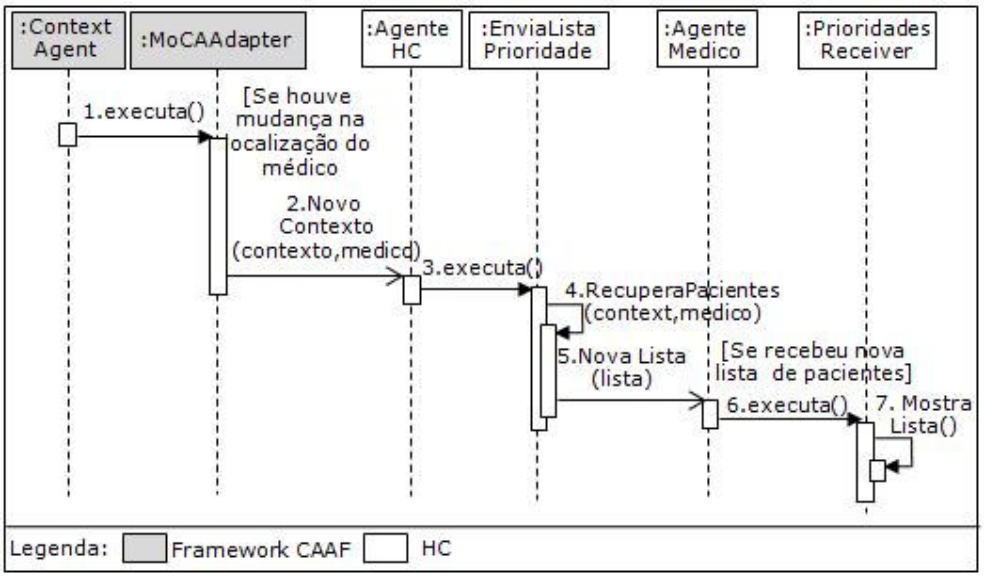

Figura 25. Cenário de Envio de Pacientes com Prioridade em HC

4.3. Análise Geral do Reuso em ASCs usando ESSMA

CAAF é um *framework* que utiliza abstrações e mecanismos de decomposição propostos por ESSMA a fim de facilitar o desenvolvimento de ASCs em conjunto com outras ferramentas existentes, como MoCA e JADE. De fato, através da instanciação da aplicação HC a partir de CAAF, vimos que o uso de agentes de software, dos elementos internos ao seu conhecimento e de suas propriedades de agência (Seção 3.1) diminui bastante a complexidade envolvida no processo de desenvolvimento de ASCs.

Por exemplo, a propriedade de *interação* de agentes de servidor e agentes de usuário, implementada em CAAF através de simples troca de mensagens `ACLMessage` de JADE (Seção 3.1.4), possibilita que usuários e serviços sejam capazes de se comunicar em um ambiente distribuído, o que é inerente às ASCs. A própria *autonomia* dos agentes, encapsulada através da classe `Agent` de JADE (Seção 3.1.4), permite que ASCs funcionem sem a intervenção direta de usuários, requisito comum em aplicações de tempo real, como é o caso HC (Seção 4.2).

Vimos também que o nível de reuso de sensibilidade ao contexto em ASCs aumenta sensivelmente, uma vez que pontos bem definidos de apenas 6 classes de CAAF precisam ser conhecidos a fim de que uma instância de ASC possa ser obtida (confira nas Figuras 23 e 24). O requisito de propagação modular de

informações de contexto é então satisfatoriamente contemplado, pois as abstrações e mecanismos propostos por ESSMA permitem definir um modelo simplificado de ASCs que enfatiza apenas os detalhes mais importantes.

Além disso, o desenvolvimento de ASCs usando CAAF foi realizado a partir de conceitos mais próximos do domínio das aplicações do que daqueles relacionados à sensibilidade ao contexto. Mais especificamente, comparando o número de classes de negócio de HC com o número de classes de CAAF usados em sua instanciação, percebemos que isto acontece, porque, em vez de propor uma solução alternativa ao uso de MoCA, CAAF é projetado de modo não só a reusar, mas, principalmente, a encapsular as funcionalidades deste *middleware*.

Portanto, pela instanciação da aplicação *Health Care*, verificamos que o *framework* CAAF torna transparente os detalhes específicos do *middleware* sendo utilizado para dar suporte à sensibilidade ao contexto em ASCs. Para desenvolver as ASCs, os engenheiros de software devem apenas conhecer as definições básicas de ESSMA (Seção 3.1) e de ASCs (Seção 2.1.1), as quais constituem os próprios conceitos utilizados na elaboração do *framework*.