5

A Arquitetura de Software ArchM

Como dito anteriormente, uma possível solução para o problema de separação de mobilidade em SMAs, investigada neste trabalho, envolve o DSOA. Nosso trabalho propõe: (1) uma arquitetura de software orientada a aspectos, denominada ArchM, que define componentes de software e suas respectivas interfaces, com o objetivo de possibilitar a separação de interesses de mobilidade em SMAs; (2) um *framework* orientado a aspectos, denominado AspectM, que implementa e valida a arquitetura ArchM. Neste capítulo, a arquitetura ArchM é descrita em detalhes. O *framework* AspectM é apresentado no Capítulo 6.

A arquitetura ArchM modulariza os interesses de mobilidade no nível de arquitetura de um sistema. Em outras palavras, no nível arquitetural, os componentes de mobilidade da arquitetura separam os interesses de mobilidade dos componentes correspondentes às funcionalidades básicas e outros interesses de SMAs. A arquitetura ArchM constitui uma extensão da arquitetura orientada a aspectos proposta por Garcia (2004), que propõe uma solução no nível arquitetural para o problema de separação de mobilidade.

Entretanto, vários problemas relativos à separação de mobilidade não foram explorados. Por exemplo, Garcia (2004) não define diretivas para a flexibilização do uso de plataformas de mobilidade em SMAs. Além disso, a arquitetura proposta por Garcia não alcança um bom nível de detalhamento na descrição dos componentes e interfaces referentes aos protocolos de mobilidade (Seção 2.2). De fato, a arquitetura de Garcia apenas introduz as questões de mobilidade que devem ser consideradas em SMAs móveis.

Neste capítulo, apresentamos as contribuições esperadas do desenvolvimento de uma arquitetura para separação dos interesses de mobilidade. Posteriormente, descrevemos a arquitetura ArchM, especificada tendo em vista este objetivo. Finalmente, apresentamos uma análise crítica das características de ArchM bem como de suas contribuições.

5.1. Uma Arquitetura de Software para Mobilidade

A utilização de uma arquitetura de software pode impactar o grau de reusabilidade e manutenibilidade no processo de desenvolvimento de sistemas (Shaw & Garlan, 1996). Por outro lado, modelos tradicionais de construção de software nem sempre funcionam bem em sistemas distribuídos, principalmente naqueles que fazem uso de mobilidade de código. Isto se deve ao fato de que as interações entre componentes arquiteturais que residem no mesmo espaço de endereçamento de uma máquina são distintas das interações que ocorrem entre componentes em diferentes máquinas de uma rede.

Portanto, a necessidade de obtenção de reusabilidade/manutenibilidade em SMAs móveis pode motivar o desenvolvimento de uma arquitetura de software para tratamento de mobilidade. Uma arquitetura deste tipo fornece um contexto no qual decisões detalhadas de projeto de mobilidade podem ser adiadas para fases posteriores do processo de desenvolvimento de software. Uma arquitetura de software para mobilidade também é independente de linguagens de programação ou de plataformas. Isto minimiza a complexidade ocasionada por problemas de entrelaçamento e espalhamento de interesses de mobilidade no código de SMAs.

Em resumo, se uma arquitetura específica para tratamento de mobilidade for considerada no desenvolvimento de SMAs móveis, há maior probabilidade de alcançar a reusabilidade e a manutenibilidade dos interesses de mobilidade, bem como de funcionalidades básicas e de outros interesses em SMAs. Na próxima seção, apresentamos nossa arquitetura de software para tratamento de mobilidade em SMAs, a arquitetura ArchM.

5.2. Componentes de ArchM

A arquitetura ArchM é composta por quatro tipos de componentes: (1) o componente `Kernel`, que modulariza as funcionalidades básicas de SMAs; (2) o componente aspectual `MobilityProtocol`, que modulariza o protocolo de mobilidade de agentes móveis em SMAs; (3) o componente aspectual `MobilityManagement`, que permite a flexibilização do uso de plataformas de mobilidade em SMAs; (4) os componentes que modularizam outros interesses adicionais, como interação, colaboração e aprendizagem. A noção de *componente aspectual* modulariza um interesse transversal no nível de arquitetura de um sistema (Capítulo 4). Desta forma, `MobilityProtocol` e `MobilityManagement` separam os interesses de mobilidade das funcionalidades básicas no componente `Kernel` e dos interesses adicionais nos outros componentes.

A Figura 20 apresenta os componentes arquiteturais de ArchM. A modelagem da arquitetura é baseada em uma extensão da linguagem ASideML (Capítulo 4). Aspectos arquiteturais são componentes representados como diamantes. Cada interface de um componente arquitetural é representada como um pequeno círculo. O relacionamento transversal é apresentado através de uma seta pontilhada, no sentido do componente aspectual para o componente afetado.

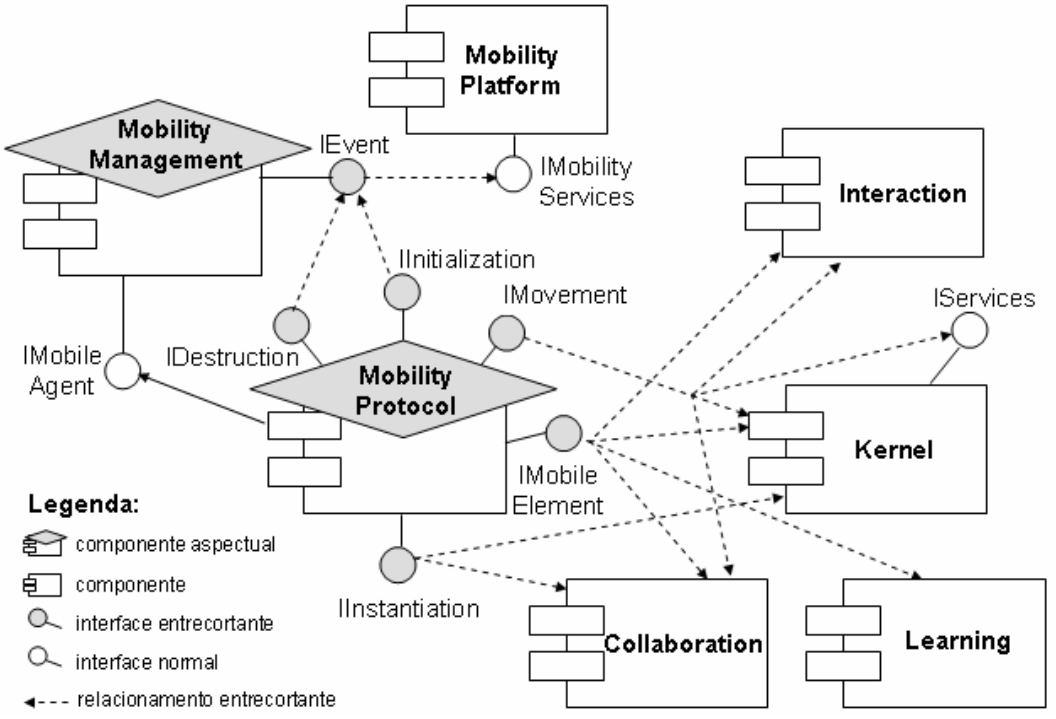

Figura 20. A Arquitetura de Software ArchM

Na arquitetura ArchM, o componente `Kernel` modulariza as funcionalidades básicas associadas a um tipo de agente. Este componente também é responsável por modularizar elementos do conhecimento intrínseco de um agente, tais como crenças, objetivos, planos e ações (Seção 1.2). Alternativamente, o componente `Kernel` pode representar um objeto estacionário que precisa ser transformado em um agente móvel. No nível de projeto, o componente `Kernel` deve ser especificado como um conjunto de classes. Além de modularizar as funcionalidades básicas e o conhecimento intrínseco de um agente, o componente `Kernel` deve implementar a interface que define os serviços fornecidos pelo agente a seus clientes. Na Figura 20, o componente `Kernel` especifica os serviços fornecidos pelo agente através da interface `IServices`.

Além do componente `Kernel`, outros componentes arquiteturais são utilizados para melhorar a separação de interesses nos SMAs móveis que seguem as diretivas de ArchM. Tais componentes modularizam propriedades como interação, colaboração e aprendizagem de agentes. Estas propriedades permanecem isoladas do conhecimento intrínseco e dos serviços de um agente, além de separadas umas das outras. No nível de projeto detalhado de SMAs, o componente arquitetural correspondente a um interesse adicional que não a mobilidade pode ser especificado como um conjunto de classes, como ocorre com o componente `Kernel`. Na Figura 20, os componentes que modularizam interesses adicionais de SMAs são representados pelos componentes correspondentes aos interesses de interação, colaboração e aprendizagem. Estes componentes implementam interfaces normais. A Figura 20 omite estas interfaces para efeitos de simplificação.

Os componentes aspectuais `MobilityProtocol` e `MobilityManagement` separam os interesses de mobilidade do componente `Kernel` e dos outros componentes arquiteturais. Nestes componentes, a mobilidade permanece isolada do conhecimento intrínseco e dos serviços de um agente, além de separada dos outros interesses adicionais nos sistemas. O objetivo é facilitar a composição e a construção de diferentes agentes com diferentes características. No nível de projeto detalhado, os componentes `MobilityProtocol` e `MobilityManagement` devem ser especificados como um conjunto de aspectos e classes auxiliares.

O componente aspectual `MobilityProtocol` tem como função principal isolar os interesses do protocolo de mobilidade de um agente (Seção 2.2); isto é, o componente `MobilityProtocol` permite especificar os interesses transversais referentes aos protocolos de instanciação, inicialização, movimentação e destruição de agentes (Seção 2.2). O componente aspectual `MobilityManagement` conecta a aplicação com o componente adicional `MobilityPlatform`. Este componente modulariza o acesso aos serviços de mobilidade disponibilizados por plataformas de agentes móveis. Além disso, o componente `MobilityManagement` possui a importante função de flexibilizar o uso destas plataformas em SMAs. Na Figura 20, note que os componentes `MobilityProtocol` e `MobilityManagement` servem em conjunto como elemento de integração entre agentes de aplicações e serviços de mobilidade providos por plataformas.

5.3. Interfaces de ArchM

Na Figura 20, o componente `MobilityManagement` possui duas interfaces. A interface normal `IMobileAgent` está relacionada à especificação de serviços providos por agentes que são recorrentes em plataformas de mobilidade. Mais especificamente, como o componente `MobilityManagement` permite a flexibilização do uso de plataformas de mobilidade, este componente disponibiliza a interface `IMobileAgent`, que permite acesso aos serviços de agentes móveis freqüentemente encontrados em SMAs também disponibilizados pelas diferentes plataformas de mobilidade.

A natureza transversal do componente `MobilityManagement` é especificada através da interface `IEvent`. Esta interface entrecorta a chamada ou a execução de serviços de plataformas de mobilidade. Por exemplo, através da interface `IEvent`, é disponibilizado o acesso à execução das etapas do ciclo de vida de agentes móveis, como a inicialização e a destruição de agentes. Isto torna possível a definição do protocolo de mobilidade sem referência a plataformas de agentes móveis específicas, diminuindo o acoplamento dos elementos arquiteturais do agente com plataformas de mobilidade em uso.

Na arquitetura ArchM, o componente aspectual `MobilityProtocol` está relacionado a mais de um componente arquitetural. Isto acontece devido à forte

natureza transversal deste componente aspectual, que, neste caso, dispõe de várias interfaces que entrecortam outros componentes da arquitetura de diferentes modos. Tais interfaces entrecortam elementos internos de um componente do agente ou as interfaces deste componente.

A interface `IMobileElement` especifica quais objetos (internos aos outros componentes arquiteturais) são móveis e quais poderão ser movidos junto com agentes. Conseqüentemente, esta interface freqüentemente entrecorta todos os outros componentes arquiteturais (Figura 20). A interface `IInstantiation` especifica quando e de que forma um representante móvel é instanciado na plataforma para um tipo ou papel de agente instanciado na aplicação. Por esta razão, a interface `IInstantiation` entrecorta tipos e papéis de agentes internos aos componentes `Kernel` e `Collaboration`, respectivamente (Figura 20). Na interface `IInstantiation`, é mantida uma associação unívoca entre um tipo ou papel móvel da aplicação e seu respectivo representante na plataforma de mobilidade. A especificação da interface `IInstantiation` possibilita a execução do protocolo de instanciação de agentes nas aplicações.

A interface `IMovement` define os componentes que disparam a movimentação do agente para ambientes remotos e o retorno deste agente para o ambiente original. Esta interface entrecorta os elementos do componente `Kernel` uma vez que o agente pode se movimentar quando elementos de seu *kernel* (planos, crenças e objetivos) sofrem mudanças. Além disso, a interface `IMovement` também entrecorta outros elementos: (1) a interface `IServices`, uma vez que a movimentação pode ocorrer antes ou após a execução de um serviço; (2) o componente `Interaction`, porque a movimentação também pode ser motivada por eventos externos ou por mensagens recebidas de outros agentes; (3) o componente `Collaboration`, uma vez que a movimentação pode ser disparada no contexto da execução de ações relacionadas ao papel de um agente; e (4) outros componentes, dependendo de características que o tipo de agente possui. A especificação da interface `IMovement` possibilita a invocação dos procedimentos de partida e a execução do protocolo de movimentação de agentes nos SMAs.

A interface `IInitialization` define os elementos que detectam a chegada do agente em um novo *host*. Por essa razão, a interface `IInitialization` entrecorta a interface `IEvent` do componente aspectual `MobilityManagement`. A

especificação da interface `IInitialization` possibilita a invocação dos procedimentos de chegada de um agente e a execução do protocolo de inicialização de agentes nas aplicações. Finalmente, a interface `IDestruction` define os elementos que detectam a destruição de um agente em uma plataforma. Assim como a interface `IInitialization`, a interface `IDestruction` entrecorta a interface `IEvent` do componente aspectual `MobilityManagement`. A especificação da interface `IDestruction` possibilita a execução do protocolo de destruição de agentes.

Note que os protocolos de instanciação, inicialização, movimentação e destruição de agentes definidos nas interfaces `IInstantiation`, `IInitialization`, `IMovement` e `IDestruction`, respectivamente, não fazem referências a plataformas de mobilidade específicas, de modo a manter a flexibilização do uso de plataformas em SMAs móveis.

5.4. Análise de ArchM

A arquitetura ArchM é independente de linguagens de programação ou plataformas de mobilidade. Isto minimiza a complexidade ocasionada por problemas de entrelaçamento e espalhamento de interesses de mobilidade no código de SMAs. As diretivas de ArchM expõem a definição de interfaces, componentes e a interação entre estes que orientam os desenvolvedores no projeto de mobilidade em SMAs móveis. Nossa arquitetura também fornece um contexto no qual decisões de projeto detalhado podem ser adiadas para fases posteriores do processo de desenvolvimento de software. Utilizamos as diretivas de ArchM durante o desenvolvimento de nosso *framework* de mobilidade (Capítulo 6).

Concluímos que a utilização da arquitetura ArchM no processo de desenvolvimento de software possibilita o aumento da reusabilidade e da manutenibilidade dos interesses de mobilidade, bem como de funcionalidades básicas e de outros interesses em SMAs móveis.