deve ser focado em questões de domínio do núcleo. Não deve ser envolvido em atividades de infra-estrutura. A interface do usuário não deve nem ser firmemente ligado à lógica de negócios, nem para as tarefas que normalmente pertencem à camada de infra-estrutura. Uma camada de aplicação é necessária em muitos casos. Tem que haver um gerente sobre a lógica de negócios que supervisiona e coordena a actividade global da aplicação.

Por exemplo, uma interação típica da aplicação, domínio e infra-estrutura poderia ser assim. O usuário quer reservar uma rota de vôos, e pede um serviço de aplicação na camada de aplicação para fazê-lo. A camada de aplicativo obtém os objetos de domínio relevantes da infra-estrutura e invoca métodos relevantes sobre eles, por exemplo, para verificar as margens de segurança para outros voos já reservados. Uma vez que os objetos de domínio fizeram todas as verificações e atualiza o seu status para “decidiu”, o serviço de aplicação persistir os objetos para a infra-estrutura.

Entidades

Há uma categoria de objetos que parecem ter uma identidade, que permanece o mesmo durante todo os estados do software. Para estes objetos não são os atributos que importa, mas um fio de continuidade e identidade, que abrange a vida de um sistema e pode se estender para além dela. Tais objetos são chamados de Entidades

linguagens OOP manter instâncias de objetos na memória, e eles associam uma referência ou um endereço de memória para cada objeto. Esta referência é única para cada objeto em um determinado momento do tempo, mas não há nenhuma garantia de que ele vai ficar assim por um período indefinido de tempo. Na verdade, o contrário é verdadeiro. Os objetos são constantemente deslocado para fora e volta para a memória, eles são serializados e enviados pela rede e recriado na outra extremidade, ou eles são destruídos. Esta referência, que se destaca como uma identidade para o ambiente de execução do programa, não é a identidade que estamos a falar. Se existe uma classe que detém tempo

informações, como a temperatura, é perfeitamente possível ter dois casos distintos da classe respectiva, ambas contendo o mesmo valor. Os objetos são perfeitamente iguais e intercambiáveis uns com os outros, mas eles têm diferentes referências. Eles não são entidades.

Se fôssemos para implementar o conceito de uma pessoa usando um programa de software, provavelmente criar uma classe Pessoa com uma série de atributos: nome, data de nascimento, local de nascimento, etc, são qualquer um desses atributos a identidade da pessoa ? O nome não pode ser a identidade porque pode haver mais pessoas com o mesmo nome. Não podia distinguir entre a pessoas com o mesmo nome, se fôssemos levar em conta apenas o seu nome. Nós não podemos usar a data de nascimento também, porque há muitas pessoas nascidas no mesmo dia. O mesmo se aplica para o lugar de nascimento. Um objeto deve ser distinguido de outros objetos mesmo que eles possam ter os mesmos atributos. confusão de identidade pode levar a corrupção de dados.

Considere um sytem contabilidade bancária. Cada conta tem o seu próprio número. Uma conta pode ser precisamente identificado pelo seu número. Este número é mantido durante toda a vida do sistema e garante a continuidade. O número da conta pode existir como um objeto na memória, ou pode ser destruído na memória e enviados para o banco de dados. Ele também pode ser arquivado quando a conta está fechada, mas ainda existe em algum lugar, desde que haja algum interesse em mantê-lo ao redor. Não importa o que a representação é preciso, o número permanece o mesmo.

Portanto, entidades de implementação em software significa a criação de identidade. Para uma pessoa que pode ser uma combinação de atributos: nome, data de nascimento, local de nascimento, nome dos pais, endereço atual. O número da Segurança Social também é usado nos EUA para criar identidade. Para uma conta bancária no número de conta parece ser suficiente para a sua identidade. Normalmente, a identidade ou é um atributo do objecto, uma combinação de atributos, um atributo especialmente criado para preservar e expressar a identidade, ou mesmo um comportamento. É importante para os dois objetos com diferentes identidades para ser para ser facilmente distinguidos pelo sistema, e dois objetos com o

mesma identidade a ser considerado o mesmo pelo sistema. Se essa condição não for atendida, em seguida, todo o sistema pode ser corrompido.

Existem maneiras diferentes de criar uma identidade única para cada objeto. O ID pode ser gerado automaticamente por um módulo, e usado internamente no software sem torná-lo visível para o usuário. Pode ser uma chave primária em uma tabela de banco de dados, o que é garantido que ser exclusivo no banco de dados. Sempre que o objeto é recuperado do banco de dados, o seu ID é recuperada e recriado na memória. O ID pode ser criado pelo usuário, como acontece com os códigos associados aos aeroportos. Cada aeroporto tem um ID de string único, que é reconhecido internacionalmente e utilizada pelas agências de viagens em todo o mundo para identificar os aeroportos em seus horários de viagem. Outra solução é usar os atributos do objeto para criar o ID, e quando isso não for suficiente, um outro atributo pode ser adicionado para ajudar a identificar o respectivo objeto.

Quando um objeto se distingue pela sua identidade, ao invés de seus atributos, fazem deste primário para a sua definição no modelo. Mantenha a simples definição de classe e focada na vida de continuidade do ciclo e identidade. Definir um meio de distinguir cada objeto independentemente da sua forma ou a história. Esteja atento aos requisitos que a chamada para a correspondência de objetos por atributos. Definir uma operação que é garantido para produzir um resultado único para cada objeto, possivelmente, anexando um símbolo que é garantido único. Este meio de identificação pode vir do lado de fora, ou pode ser um identificador arbitrário criado por e para o sistema, mas devem corresponder aos distinções de identidade no modelo. O modelo deve definir o que significa ser a mesma coisa.

Entidades são objetos importantes de um modelo de domínio, e eles devem ser considerados desde o início do processo de modelagem. Também é importante para determinar se um objeto precisa ser uma entidade ou não, o que é discutido no próximo padrão.