4
Visão geral da arquitetura para OnOCs

4.1
Motivação

Com a difusão do uso de microcomputadores, a economia mundial torna-se cada vez mais dependente da tecnologia da informação para realização de seus negócios. À medida que os recursos oferecidos evoluem, a disseminação do uso da tecnologia da informação também aumenta. Deste modo, torna-se necessário uma constante evolução na produção de software a fim de permitir que as novas necessidades da tecnologia da informação sejam atendidas satisfatoriamente, com agilidade e eficácia.

Como a tarefa de programação tem sido sempre considerada difícil, a área de Engenharia de Software vem propondo metodologias e padrões de desenvolvimento, e fornecendo ferramentas e *frameworks*, de forma a facilitar o projeto e implementação de sistemas. Portanto, para facilitar e agilizar o desenvolvimento de software é necessário especificar detalhadamente a organização e a estrutura geral do sistema, ou seja, especificar a arquitetura do software.

A palavra arquitetura está associada a arte e ciência de projetar e construir, levando em consideração os métodos, ferramentas e padrões utilizados no projeto e na construção. Uma arquitetura de software define a estrutura dos componentes de um sistema e seus inter-relacionamentos, e os princípios e guias para desenvolvimento e evolução do sistema (Garlan & Perry, 1995).

O capítulo anterior abordou a metodologia utilizada para viabilizar a interoperabilidade entre fontes de objetos usando um catálogo de objetos baseado em ontologias (OnOC). Este capítulo introduz a arquitetura para criação de OnOCs descrevendo os módulos necessários e os serviços essenciais para cobrir os requisitos levantados no capítulo anterior. A definição de uma arquitetura para OnOCs visa permitir um fácil entendimento para criação de OnOCs.

PUC-Rio - Certificação Digital Nº 0310880/CB

4.2
Componentes da arquitetura

Esta seção apresenta a estrutura de componentes da arquitetura proposta neste trabalho para o desenvolvimento de catálogos de objetos baseados em ontologias.

A Figura 29 mostra a organização em módulos da arquitetura proposta para um OnOC. A seguir são detalhados os módulos componentes desta arquitetura.

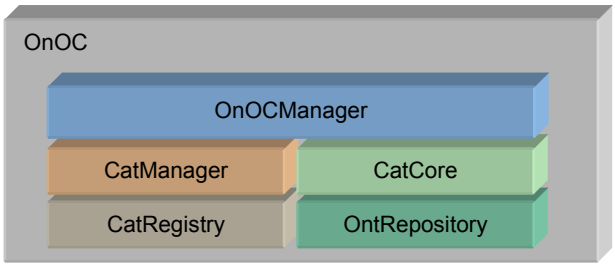

Figura 29 – Arquitetura em módulos para OnOCs.

O módulo *OnOCManager* representa a interface de acesso às operações do catálogo. Este módulo é responsável por atender as operações acessadas pelo módulo de fornecimento dos serviços aos clientes do catálogo, sejam eles humanos ou outros sistemas. Os serviços do OnOC podem ser fornecidos aos clientes via um protocolo específico, como por exemplo: HTTP ou CORBA, ou até mesmo utilizando uma interface que implemente serviços web (*Web Services*).

CatManager é o módulo de gerenciamento do catálogo (*Catalog Manager*). É o módulo responsável pelas tarefas de gerenciamento do ambiente. A comunicação com este módulo é feita através da interface *CatManagerFacade*. As tarefas do módulo *CatManager* subdividem-se em: gestão de acessibilidade e gestão de cadastramento, executadas por dois módulos distintos, como mostra a Figura 30.

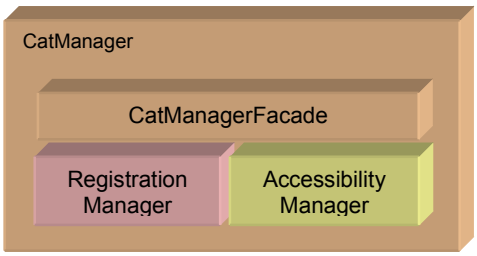

Figura 30 – Subdivisão do módulo *CatManager*.

PUC-Rio - Certificação Digital Nº 0310880/CB

O *RegistrationManager* é o módulo responsável pelo cadastramento das aplicações participantes e das fontes de objetos. O módulo *AccessibilityManager* faz o controle e cadastramento das permissões de acesso das aplicações participantes às ontologias mantidas no catálogo.

O módulo *CatRegistry*, mostrado na Figura 31, é o módulo responsável pelo repositório do catálogo onde são cadastradas as informações manipuladas pelo módulo *CatManager*. O repositório gerenciado pelo módulo *CatRegistry* mantém informações a respeito dos usuários, fontes de objetos e permissões. Toda comunicação com o repositório é feita através da interface *CatRegistryFacade*. Este módulo comunica-se com o banco de dados através de uma interface *DatabaseConnector*, como por exemplo ODBC (*Open DataBase Connectivity*) ou JDBC (*Java Database Connectivity*), permitindo a comunicação com qualquer sistema gerenciador de banco de dados (SGBD) para armazenamento destas informações.

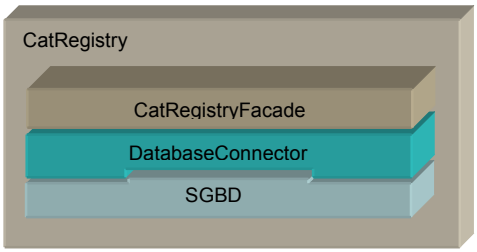

Figura 31 – Organização do módulo *CatRegistry*.

O módulo responsável pela manipulação e consulta às ontologias armazenadas no catálogo é chamado *OntManager* (vide Figura 29). Este módulo também é responsável pela manipulação e consulta aos mapeamentos entre as ontologias locais e a ontologia de referência do catálogo. As tarefas deste módulo são realizadas em quatro sub-módulos, como mostra a Figura 32. A comunicação com o módulo *OntManager* é feita através da interface *OntManagerFacade*.

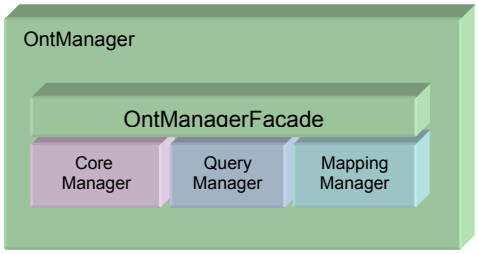

Figura 32 – Subdivisão do módulo *OntManager*.

PUC-Rio - Certificação Digital Nº 0310880/CB

O módulo *CoreManager* é responsável pela manipulação das ontologias armazenadas no OnOC. Através deste módulo as ontologias são armazenadas e recuperadas do repositório de ontologias. O módulo *QueryManager* é o módulo responsável por consultar as ontologias mantidas pelo OnOC. De acordo com a situação de uso do catálogo, este módulo pode ser implementado de forma a satisfazer os requisitos de consulta. Um exemplo disso é a utilização do OnOC como mediador. O módulo *MappingManager* é responsável pelos mapeamentos entre as ontologias locais e a ontologia de referência. Este módulo pode fazer uso de uma máquina de inferência para processar as consultas.

O módulo *OntRepository*, mostrado na Figura 29, é responsável pelo repositório de ontologias do OnOC. Neste repositório são armazenadas as ontologias locais, a ontologia de referência do catálogo e os mapeamentos entre as ontologias locais e a ontologia de referência. Como mostra a Figura 33, a *OntRepositoryFacade* é a interface definida para comunicação do repositório com os demais módulos da arquitetura. Para viabilizar a utilização de qualquer tecnologia para armazenamento de ontologias, como por exemplo Jena ou Sesame, este módulo oferece uma classe de adaptação chamada *OntRepositoryAdapter* para fazer a camada de adaptação do catálogo com o repositório utilizado, representado pelo módulo *OntologyRepository.* A Figura 33 mostra a arquitetura genérica do *OntRepository* onde o *OntologyRepository* representa um repositório genérico que pode ser implementado usando um repositório Jena, Sesame, ou outro. A escolha do repositório fica a cargo do desenvolvedor.

A Figura 34 mostra a arquitetura detalhada de um OnOC em módulos e sub-módulos.

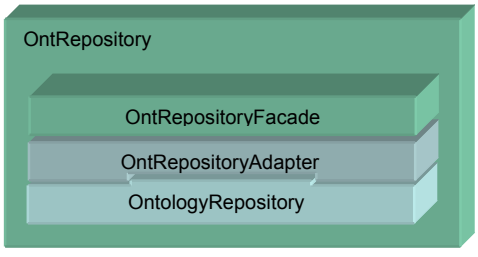

Figura 33 – Subdivisão do módulo *OntRepository*.

PUC-Rio - Certificação Digital Nº 0310880/CB

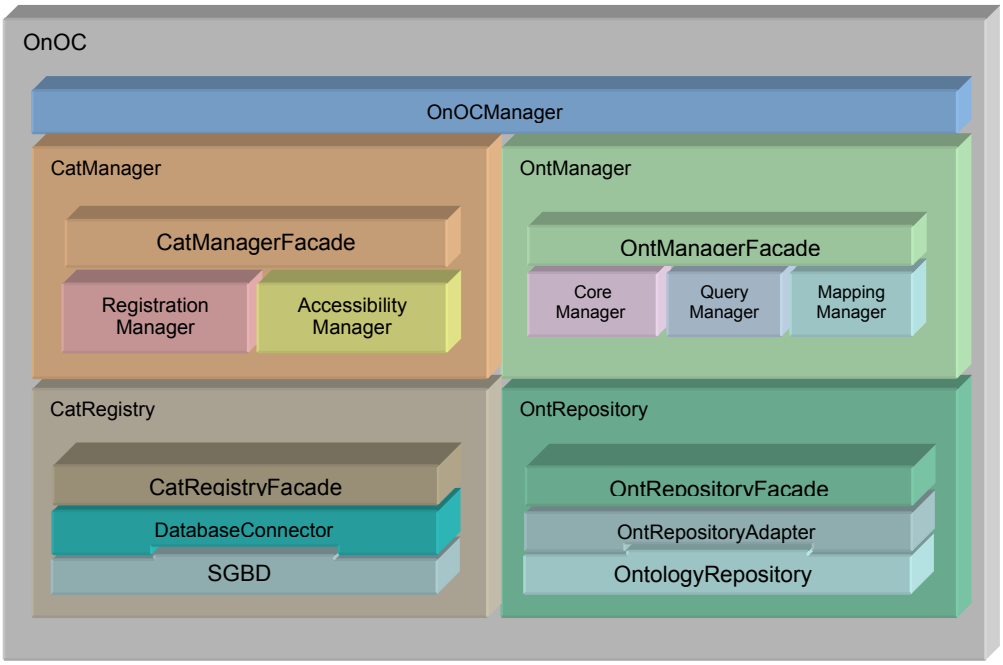

Figura 34 – Arquitetura detalhada para OnOCs.

4.3
Inter-relacionamentos da arquitetura

Esta seção apresenta os inter-relacionamentos entre os módulos componentes da arquitetura proposta. Os inter-relacionamentos entre os componentes da arquitetura evidenciam a comunicação entre os módulos do sistema que será criado a partir desta arquitetura. A identificação destes inter-relacionamentos facilita o entendimento do comportamento do sistema gerado.

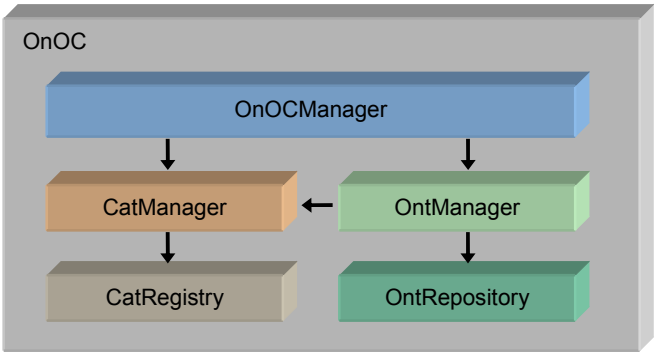

Figura 35 – Primeiro nível de inter-relacionamentos entre os módulos da arquitetura.

PUC-Rio - Certificação Digital Nº 0310880/CB

Basicamente, os módulos comunicam-se conforme mostrado na Figura 35. A interface *OnOCManager* comunica-se com as interfaces dos módulos *CatManager* e *OntManager*. Estes, por sua vez, comunicam-se respectivamente com os módulos *CatRegistry* e *OntRepository*. Além disso, os módulos *CatManager* e *OntManager* comunicam-se entre si para compartilharem as informações mantidas pelo *CatRegistry*.

Os exemplos das Figuras 36 e 37 a seguir evidenciam os inter-relacionamentos entre os módulos da arquitetura de um OnOC.

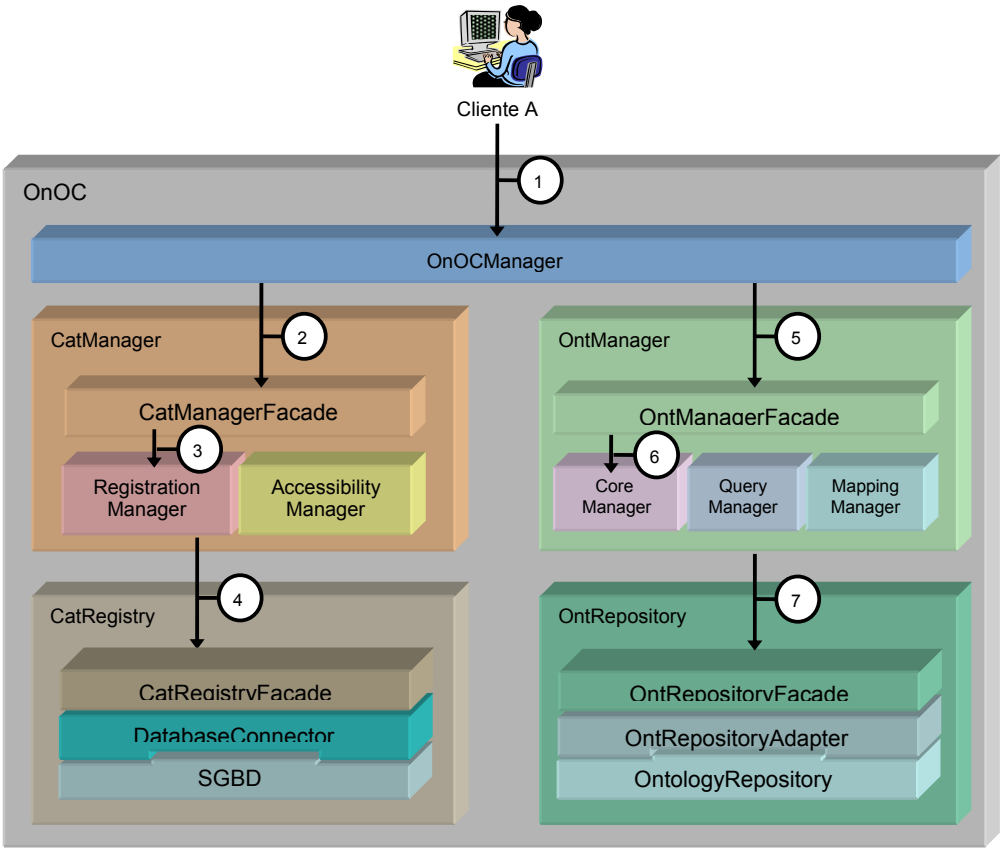

Figura 36 – Detalhamento dos inter-relacionamentos da arquitetura – Descreve termo.

Na Figura 36 podem ser vistos os passos executados por um OnOC para responder uma requisição por um termo de uma ontologia. O primeiro passo, identificado pelo número 1 na Figura 36, é a requisição do cliente ao sistema. O cliente solicita ao catálogo a descrição de um termo de uma ontologia, dada sua URI. O sistema recebe a requisição através da interface *OnOCManager*. Esta, por sua vez, comunica-se com o módulo *CatManager* (número 2) para verificar a existência do nome de usuário e senha cadastrados no registro do catálogo.

PUC-Rio - Certificação Digital Nº 0310880/CB

Para isso, a requisição é enviada ao sub-módulo *AccessibilityManager* (número 3). Este envia a consulta ao módulo *CatRegistry*, responsável pela consulta no banco de dados do catálogo (número 4). Se conferem as informações do cliente, a requisição é enviada do módulo *OnOCManager* para o módulo *OntManager* (número 5). Este, comunica-se via a interface *OntManagerFacade* com o módulo *CoreManager* para verificar se o termo esta catalogado (número 6). Se confirmado, a requisição é encaminhada para o sub-módulo *OntRepository* (número 7), que por sua vez, consulta o repositório de ontologias em busca da descrição do termo requisitado.

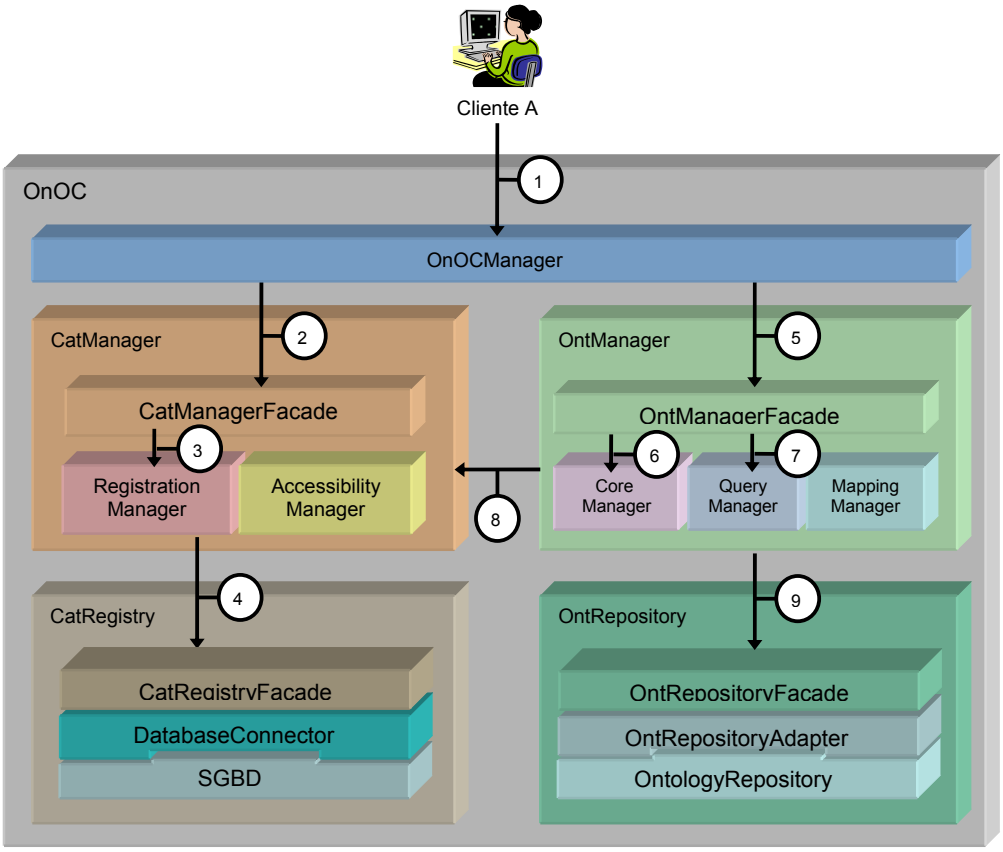

Figura 37 – Detalhamento dos inter-relacionamentos da arquitetura - Consulta.

Na Figura 38 podem ser vistos os passos executados por um OnOC para responder uma requisição de consulta de um cliente. O primeiro passo, identificado pelo número 1 na Figura 38, é a requisição do cliente ao sistema. O cliente envia a consulta ao catálogo. O sistema recebe a consulta através da interface *OnOCManager*. Esta, por sua vez, comunica-se com o módulo *CatManager* (número 2) para verificar a existência do nome de usuário e senha

PUC-Rio - Certificação Digital Nº 0310880/CB

cadastrados no registro do catálogo. Para isso, a requisição é enviada ao sub-módulo *AccessibilityManager* (número 3). Este envia a consulta ao módulo *CatRegistry*, responsável pela consulta no banco de dados do catálogo (número 4). Se conferem as informações do cliente, a requisição de consulta é enviada do módulo *OnOCManager* para o módulo *OntManager* (número 5). Este, comunica-se via a interface *OntManagerFacade* com o módulo *CoreManager* para verificar se a ontologia solicitada está catalogada (número 6). Se confirmada, a consulta é encaminhada para o sub-módulo *QueryManager* (número 7). Se a consulta solicitada não fizer uso da máquina de inferência o passo número 8 é descartado e a consulta é passada diretamente para o módulo *OntRepository* (número 9). Se a consulta fizer uso da máquina de inferência, o módulo *QueryManager* solicita a URI da ontologia de referência ao módulo *CatManager* (número 8) e então solicita ao módulo *OntRepository* (número 9) a ontologia local utilizada na consulta e a ontologia de referência. O módulo *QueryManager*, de posse das ontologias, usa a máquina de inferência para gerar o modelo de inferência utilizado para realizar a consulta do usuário. O resultado da consulta é retornado ao usuário.

Os passos executados para atender às requisições desta e outras operações do catálogo podem ser vistos em mais detalhes nos diagramas de seqüência da arquitetura (vide seção 5.5).

4.4 Padrões de software utilizados

Os padrões na engenharia de software servem para documentar soluções para diferentes tipos de problemas que ocorrem ao longo do processo de desenvolvimento de software. O objetivo é permitir a reutilização destas soluções por outros desenvolvedores ao confrontarem-se com problemas similares.

No desenvolvimento de sistemas os padrões de software podem ser usados em diferentes níveis de abstração. Entre outros, existem os padrões de projeto, padrões arquiteturais, padrões de análise e os padrões de código.

Os padrões de software tornaram-se popular através da referência (Gamma et. al., 1995). Porém, diversos padrões vêm sendo definidos para diferentes domínios, tais como: processos de desenvolvimento, arquitetura e projeto de software, entre outros.

Em (Buschmann et al., 1996), os padrões são classificados em três tipos: padrões arquiteturais, padrões de projeto e idioma. As seções subseqüentes

PUC-Rio - Certificação Digital Nº 0310880/CB

identificam os padrões arquiteturais e de projeto utilizados para definição da arquitetura proposta neste trabalho. O objetivo de uso de padrões nesta arquitetura é fornecer modelos reusáveis para o desenvolvimento de catálogos de objetos baseados em ontologias.

4.4.1 Padrão arquitetural

Um padrão para arquitetura de software descreve uma solução genérica para problemas de design recorrentes. Embora um padrão determine a estrutura básica para solução de um problema específico de projeto, ele não especifica uma solução detalhada e completa. Um padrão fornece um esquema genérico para uma solução de uma família de problemas. O engenheiro deve implementar este esquema de acordo com as necessidades específicas do problema que possui.

Os padrões arquiteturais expressam a organização estrutural fundamental para sistemas de software ou hardware. Um padrão arquitetural fornece um conjunto de subsistemas (componentes) predefinidos, a especificação de suas responsabilidades, e inclui regras e diretrizes para organizar os relacionamentos entre eles (inter-relacionamentos) (Buschmann et al., 1996).

Esta seção apresenta o padrão arquitetural utilizado na especificação da arquitetura proposta nesta dissertação: o padrão em camadas.

Segundo Buschmann et al. (1996), o padrão em camadas ajuda a estruturar aplicações que podem ser decompostas em grupos de sub-tarefas, onde cada grupo de sub-tarefas pertencerá a um nível de abstração.

Usando o padrão arquitetural em camadas, o sistema fica estruturado em níveis sobrepostos. O nível mais baixo de abstração é na camada de base da arquitetura. As demais camadas aumentam o nível de abstração gradativamente até a arquitetura atingir o nível de funcionalidade desejável do sistema. Em cada camada da arquitetura são organizados os componentes que operam no mesmo nível de abstração. Nesta arquitetura um serviço fornecido por uma camada é formado pela composição dos serviços fornecidos pela camada localizada um nível abaixo. Além disso, um serviço pode depender de outros serviços da mesma camada. Portanto, em uma arquitetura em camadas uma camada X fornece serviços que são utilizados pela camada X+1 e delega sub-tarefas à camada X-1.

PUC-Rio - Certificação Digital Nº 0310880/CB

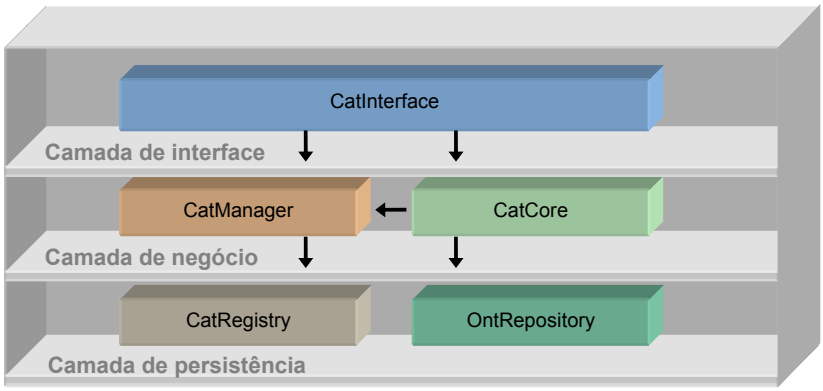

Figura 38 – Arquitetura usando o padrão em camadas.

A Figura 38 mostra a arquitetura proposta nesta dissertação organizada utilizando o padrão arquitetural em camadas.

Na camada de interface é feita a comunicação com os clientes da aplicação. Nesta camada são fornecidos os serviços do catálogo aos usuários, sejam estes humanos, agentes de software ou outras aplicações. Os serviços da camada de interface são compostos de operações fornecidas pela camada de negócios.

A camada de negócios implementa as operações que compõem os serviços fornecidos pelo catálogo na camada de interface. Esta camada é sub-dividida em módulos que operam num mesmo nível de abstração. A camada de negócios recebe as solicitações da camada de interface e transforma-as em requisições à camada de persistência.

Na camada de persistência são armazenadas e recuperadas as informações manipuladas pela camada de negócios. A camada de persistência é composta por módulos. Cada módulo faz acesso a um repositório de informações. As solicitações da camada de negócios são recebidas pela camada de persistência que as executa e devolve o resultado à camada de negócios, que por sua vez, manipula o resultado para devolvê-lo à camada de interface.

4.4.2 Padrões de projeto

O paradigma de programação orientado a objetos surgiu para resolver o problema de reaproveitamento e manutenção de código. Porém, foi comprovado que o simples uso da orientação a objetos não garante que obtenhamos sistemas extensíveis e reutilizáveis. De fato, o reuso não se obtém do simples

PUC-Rio - Certificação Digital Nº 0310880/CB

reaproveitamento de código, e toda experiência obtida no desenvolvimento de software é difícil de ser compartilhada.

Portanto, o projeto do software é a forma que um engenheiro tem de documentar o conhecimento a respeito da solução de determinado problema. Para auxiliar nesta tarefa foram criados os padrões de projeto. O uso de padrões permite que os engenheiros de software compartilhem experiências bem sucedidas na resolução de problemas recorrentes no desenvolvimento. De fato, os padrões de projeto formam um vocabulário comum para discussão de questões relativas ao projeto de software.

Um padrão de projeto captura a essência de um problema de projeto genérico e propõe uma solução em um contexto específico (Gamma et al., 1995). A solução proposta por um padrão de projeto deve ser entendida por outros desenvolvedores de forma a ser reutilizada em situações similares.

Em Gamma et al. (1995), os padrões de projeto são classificados seguindo dois critérios, como mostra a Tabela 3. O primeiro critério, chamado *propósito*, está relacionado com a finalidade do padrão, que pode ser de três tipos: criação, estrutural ou comportamental. O segundo critério, chamado *escopo*, indica se o padrão é aplicado às classes ou aos objetos.

Os padrões de criação preocupam-se com o processo de criação dos objetos. Os padrões estruturais lidam com a composição de classes ou objetos. Os padrões comportamentais descrevem a forma na qual as classes ou os objetos interagem e distribuem as responsabilidades.

Tabela 6 – Classificação dos padrões de projeto (Gamma et al., 1995).

		PROPÓSITO		
		Criação	**Estrutural**	**Comportamental**
ESCOPO	**Classe**	*Factory Method*	*Adapter (class)*	*Interpreter* *Template Method*
	Objeto	*Abstract Factory* *Builder* *Prototype* *Singleton*	*Adapter (object)* *Bridge* *Composite* *Decorator* *Facade* *Proxy*	*Chain of Responsibility* *Command* *Iterator* *Mediator* *Memento* *Flyweight* *Observer* *State* *Strategy* *Visitor*

As seções seguintes apresentam os padrões de projeto sugeridos para criação de catálogos seguindo a arquitetura proposta nesta dissertação.

PUC-Rio - Certificação Digital Nº 0310880/CB

4.4.2.1
Padrão *Facade*

O *Facade* é um padrão de projeto estrutural utilizado para fornecer uma interface unificada (fachada) para os módulos de um sistema. O objetivo deste padrão é definir uma interface de alto nível para facilitar o uso das funcionalidades do módulo.

A arquitetura proposta nesta dissertação está organizada em camadas compostas por módulos. Um objetivo comum entre projetos orientados a objetos é diminuir a comunicação e as dependências entre as classes do sistema (reduzir o acoplamento), de forma a facilitar a manutenção do sistema. Suponha a situação de alto acoplamento mostrada pela Figura 39 (a), onde os módulos X, Y e Z acessam diretamente as classes A_1, A_2 e A_3 do módulo A. Na Figura 39 (b) é possível perceber a redução do acoplamento através do uso do padrão *Facade*.

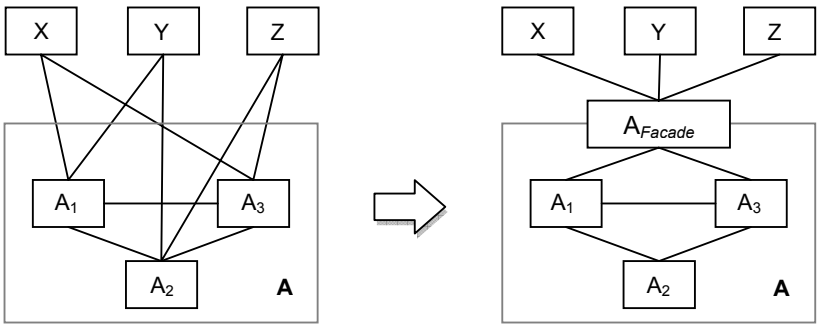

Figura 39 – Uso do padrão de projeto *Facade.*

Na arquitetura proposta nesta dissertação, o padrão *Facade* é usado como interface de comunicação de cada módulo componente da arquitetura. Portanto, toda comunicação entre os módulos de um OnOC baseado nesta arquitetura é feita através de suas fachadas.

Para garantir que uma fachada seja única e acessível de forma global, o padrão *Singleton* é utilizado em conjunto com o padrão *Facade.*

4.4.2.2
Padrão *Singleton*

O padrão *Singleton* é um padrão de projeto de criação de objetos. O objetivo é garantir uma instância única e acessível de forma global e uniforme

PUC-Rio - Certificação Digital Nº 0310880/CB

para toda classe que implemente este padrão. Neste contexto, este padrão foi usado para tornar a interface de cada módulo da arquitetura única e acessível de forma global.

Outra forma de garantir o acesso global é o uso de uma variável global. Porém a variável global não garante a unicidade. A solução proposta pelo padrão *Singleton* é tornar a própria classe responsável pela sua única instância. A classe que implementa o padrão *Singleton* garante o acesso à sua instância e ainda intercepta as requisições para criação de novos objetos, garantindo que nenhuma outra instância seja criada.

4.4.2.3 Padrão *Adapter*

Uma forma de garantir o uso de qualquer tecnologia como repositório de ontologias da arquitetura proposta, foi utilizado o padrão de criação *Adapter*.

O padrão *Adapter* tem o objetivo de viabilizar a comunicação entre classes com interfaces incompatíveis. O padrão *Adapter* é responsável pela conversão de uma interface em outra.

Na Figura 40 é apresentada a estrutura para implementação do padrão *Adapter*. A classe *Target* representa a interface específica de domínio utilizada pelo cliente. A classe *Adaptee* representa a interface existente que precisa ser adaptada. A classe *Adapter* faz a adaptação da interface *Adaptee* para a interface *Target*. Os clientes (*Client*) chamam as operações da classe *Adapter*, que por sua vez, chamam as operações do *Adaptee* para atenderem as requisições do cliente.

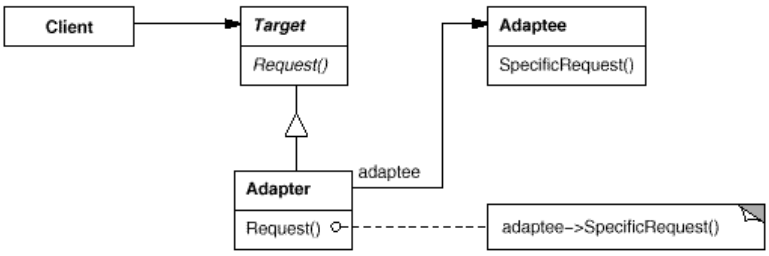

Figura 40 – Estrutura do padrão *Adapter* (Gamma et al., 1995).

PUC-Rio - Certificação Digital Nº 0310880/CB

4.5 Serviços da arquitetura

Esta seção descreve os serviços oferecidos pela arquitetura introduzida neste capítulo. Os serviços foram derivados a partir dos requisitos para OnOCs descritos no Capítulo 3.

A Tabela 7 mostra os serviços essenciais que devem ser oferecidos pela camada de interface dos sistemas que seguem a arquitetura para OnOCs descrita neste capítulo. A Tabela 7 apresenta o nome do serviço, uma breve descrição de sua funcionalidade e os requisitos atendidos. Os números identificadores dos requisitos remetem à seção 3.3.3 do Capítulo 3.

Tabela 7 – Serviços oferecidos pela camada de interface.

Serviço	Funcionalidade	Requisito(s) atendido(s)
(S1) Registrar aplicação	Cadastramento de novos clientes do catálogo	(R1)
(S2) Definir permissão	Gerenciamento das permissões de acesso das aplicações às ontologias do catálogo	(R2)
(S3) Definir administrador	Definição do usuário administrador do catálogo	(R2)
(S4) Autenticar cliente	Autenticação de clientes para acesso ao catálogo	(R3)
(S5) Definir ontologia de referência	Definição da ontologia de referência do catálogo	(R4)
(S6) Registrar fonte de objetos	Cadastramento de novas fontes de objetos	(R4)
(S7) Consultar ontologia	Consulta uma ontologia do catálogo	(R5) (R6) (R7)
(S8) Consultar classe	Consulta uma classe de uma ontologia	(R5)
(S9) Consultar propriedade	Consulta uma propriedade de uma ontologia	(R5)
(S10) Consultar instância	Descrição de uma instância de uma ontologia	(R6) (R7)
(S11) Consultar catálogo	Consulta o catálogo usando uma linguagem de consulta	(R5) (R6) (R7)

PUC-Rio - Certificação Digital Nº 0310880/CB