2.1.1.1 Modelo OpenCOM

O OpenCOM é um modelo de componentes de baixo peso projetado para o desenvolvimento de middlewares em dispositivos de computação com poucos recursos (processamento, memória, armazenamento). Além de ser um modelo de baixo peso, o OpenCOM provê a capacidade de reconfiguração dinâmica de middlewares tanto no domínio estrutural quanto no comportamental {rocha2008}.

O OpenCOM é fundamentado em três tecnologias {nascimento2013}:

- Componentes - O modelo permite a especificação da estrutura de sistemas através do uso de componentes e conexões entre componentes (ROCHA, 2008).

Os conceitos fundamentais no OpenCOM são interfaces, receptáculos e conexões. Uma interface representa uma unidade de provisão de serviços, enquanto que um receptáculo representa uma unidade de requerimento de serviços e é usado para tornar explícita a dependência de uma interface de um componente com outra. Uma conexão representa uma ligação entre um serviço fornecido por uma interface de um componente e um serviço requerido por um receptáculo de outro componente {clarke2001};

- Reflexão computacional - O modelo OpenCOM foi projetado para suportar a reflexão computacional - que é a capacidade que um sistema tem de observar sua própria representação/estrutura e modifica-la em tempo de execução;

- Frameworks de componentes – Uma característica chave do OpenCOM é o uso da noção de frameworks de componentes. Um framework de componentes é definido no OpenCOM como um conjunto fortemente acoplado de componentes que coopera para resolver alguma área de interesse. O OpenCOM também fornece um protocolo de extensão bem definido para a aceitação de componentes adicionais que modificam ou estendem o comportamento do framework de componentes, além de restringir o modo como os componentes são organizados {coulson2008}.

Cada componente OpenCOM implementa quatro interfaces, além de interfaces personalizadas, como mostra a Figura {002opencom.png} {grace2007 apud nascimento2013}:

- ILifeCycle - fornece as operações para a inicialização e a finalização do ciclo de vida de um componente;

- IConnections (opcional) - oferece os métodos para modificar as interfaces ligadas aos receptáculos de um componente. Esta interface deve ser implementada por todos os componentes que possuem receptáculo;

- IMetaInterface – suporta a inspeção dos tipos de interfaces e receptáculos declarados pelo componente;

- IUnknown - é equivalente à interface do mesmo nome no Microsoft COM, isto é, é usada para obter a referência para a interface solicitada na instância do componente.

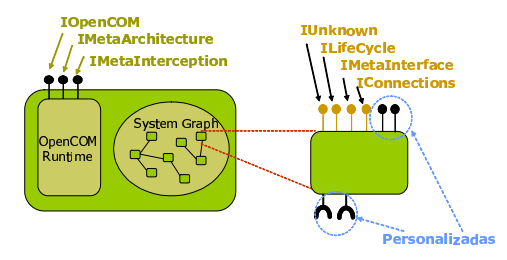


Figura {figuras/002opencom.png} – Interfaces dos metamodelos do OpenCOM {rocha2008}

O OpenCOM implanta um substrato padrão em *runtime* que está disponível em todo o espaço de endereçamento. Insto é implementado através de um componente *singleton* chamado "OpenCOM” que exporta uma interface chamada IOpenCOM. O papel do *runtime* do OpenCOM é o de gerir um repositório de componentes disponíveis e, assim, permitir a criação e exclusão de componentes. Além disso, a interface IOpenCOM serve como um ponto centralizado para a submissão de todas as solicitações de conexão ou desconexão entre receptáculos e interfaces no seu espaço de endereçamento. Para facilitar a reconfiguração, o *runtime* registra cada criação e exclusão de cada componente ou conexão em um espaço de meta-estrutura chamado de *system graph* (grafo do sistema). Isto permite que o OpenCOM suporte consultas que, através de um identificador de conexão, fornece detalhes sobre o receptáculo e as interfaces participantes da conexão, juntamente com detalhes de seus componentes que os implementam {clarke2001}.

2.1.1.2 Modelo Fractal

O Fractal é definido em {bruneton2006} como um modelo de componentes geral e extensível, projetado para implementar, implantar e gerenciar sistemas de *software* complexos, incluindo, em particular, sistemas operacionais e *middlewares*. As principais motivações do modelo são: (i) Composição de componentes, onde um componente pode conter outros componentes, permitindo uma visão uniforme das aplicações em vários níveis de abstração; (ii) Compartilhamento de componentes entre estruturas compostas de componentes, como forma de compartilhar recursos enquanto se mantém o encapsulamento de um componente; (iii) Capacidades reflexivas, para monitorar e controlar um sistema em execução; (iv) Capacidades de reconfiguração, como forma de implantar e configurar dinamicamente um sistema.

De acordo com {coupaye2007}, o modelo de componentes Fractal suporta várias linguagens de programação, como por exemplo, Java e C, e de forma experimental .NET, SmallTalk, Python e C++.

No contexto do Fractal os **componentes** são entidades de *runtime* que estão em conformidade com o modelo, estão encapsuladas, possuem identificações únicas e suportam uma ou mais interfaces. As **interfaces** são os pontos únicos de interação entre os componentes e expressam a dependência desses em termos de interfaces requeridas e providas. Os ***bindings*** são os canais de comunicação entre as interfaces dos componentes {coupaye2007} {bruneton2006}.

Um componente Fractal é a composição de uma **membrana** e um **conteúdo**, como pode ser observado na Figura {figuras/004fractal.png}. A membrana tem o papel de fornecer interfaces para um controle reflexivo sobre o conteúdo. O conteúdo consiste num conjunto finito de outros subcomponentes. A membrana de um componente pode ter interfaces internas, acessíveis somente pelos subcomponentes internos, e externas, acessíveis de fora do componente. Além disso, uma membrana possui diversas interfaces de controle, que atuam como interceptadores entre as operações de chamada que entram e saem do componente, e adicionam comportamentos aos manipuladores de tais operações {bruneton2007}.

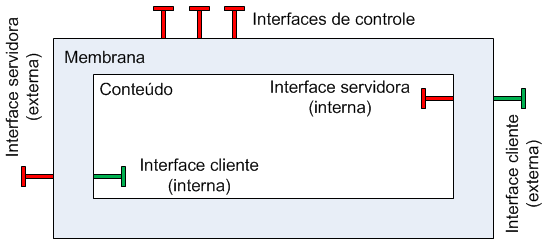


Figura {figuras/004fractal.png} – Estrutura de um componente Fractal {bruneton2006 apud nascimento2013}.

2.1.1.3 Modelo OSGi

Segundo {osgi}, o OSGi é um conjunto de especificações que definem um sistema de componentes dinâmico para o Java. Com essas especificações é possível criar um sistema composto dinamicamente por diversos componentes reusáveis. O OSGi permite que os componentes escondam suas implementações de outros componentes enquanto se comunicam através de **serviços**. Os serviços são objetos compartilhados de maneira especifica entre componentes.

A arquitetura do OSGi é composta por camadas como mostra a Figura {figuras/005osgi.png}. Elas são brevemente descritas a seguir {osgi}:

- *Bundles* – OS *bundles* são os componentes OSGi implementados pelos desenvolvedores.

- *Services* – A camada de Serviços conecta os *bundles* de maneira dinâmica. Os serviços são publicados pelos *bundles*, e são passíveis de busca e conexão posteriormente.

- *Life-Cycle* – Parte da API do OSGi que permite a instalar, desinstalar, executar, parar, e atualizar *bundles*.

- *Modules* – Camada que define como um *bundle* pode importar e exportar código.

- *Security* – Camada que manipula os aspectos de segurança do OSGi.

- *Execution Environment* – É o ambiente de execução. Define quais métodos e classes estarão disponíveis na plataforma específica.

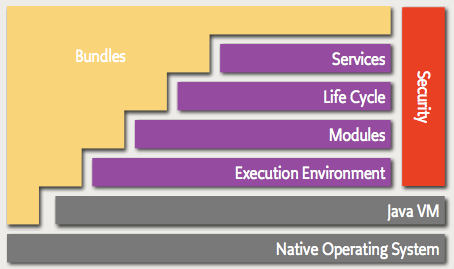


Figura {figuras/005osgi.png} – Camadas da arquitetura OSGi {osgi}.

O OSGi é apenas uma especificação de um framework para o desenvolvimento de aplicações modulares. Existem diversas implementações dessa tecnologia, sendo as mais conhecidas: Equinox, Felix e Knopflerfish. A primeira é uma implementação da especificação OSGi desenvolvida pela Eclipse Foundation. Ela é utilizada em diversas aplicações, inclusive na IDE Eclipse. Já a implementação Felix é desenvolvida e mantida pela Apache Software Foundation. A implementação Knopflerfish é desenvolvida e mantida pela Makewave.

2.9 Discussão