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:

- Componentes - O modelo permite a especificação da estrutura de sistemas através do uso de componentes e conexões entre componentes (ROCHA, 2008). Cada componente pode possuir um conjunto de interfaces (ou interfaces providas) - através das quais o componente exporta seus serviços - e um conjunto de receptáculos (ou interfaces requeridas) - através dos quais o componente requisita serviços de outros componentes;

- Reflexão computacional - O modelo foi projetado para dar apoio à reflexão computacional - que é a capacidade que um sistema tem de observar sua própria representação/estrutura ou até mesmo modificá-la;

- Frameworks de componentes - O modelo também permite a criação de frameworks de componentes. Um framework de componentes é definido no OpenCOM como uma subarquitetura de componentes sobre a qual pode ser checado um conjunto de propriedades arquiteturais desejadas (ROCHA, 2008). Essa subarquitetura também pode ser vista como um componente complexo que é composto de outros subcomponentes interconectados.

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}

Cada componente OpenCOM implementa quatro interfaces (GRACE, 2007):

- ILifeCycle - fornece operações de inicialização e finalização que são chamadas quando um componente é criado ou destruído, ou seja, define ações que devem ser tomadas no início e no final do ciclo de vida do componente;

- IConnections (opcional) - oferece 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 inspeção dos tipos de interfaces e receptáculos declarados pelo componente;

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

2.1.1.2 Modelo Fractal

2.1.1.3 Modelo OSGI

2.9 Discussão

@incollection{councill2001,

author = {Councill, Bill and Heineman, George T.},

chapter = {Definition of a Software Component and Its Elements},

title = {Component-based Software Engineering},

editor = {Heineman, George T. and Councill, William T.},

year = {2001},

isbn = {0-201-70485-4},

pages = {5--19},

numpages = {15},

url = {http://dl.acm.org/citation.cfm?id=379381.379438},

acmid = {379438},

publisher = {Addison-Wesley Longman Publishing Co., Inc.},

address = {Boston, MA, USA},

}

@book{sommerville2011,

title={ENGENHARIA DE SOFTWARE},

author={SOMMERVILLE, IAN},

isbn={9788579361081},

edition={9 ed.},

url={http://books.google.com.br/books?id=H4u5ygAACAAJ},

publisher={PEARSON BRASIL},

address={São Paulo, SP, Brasil}

}

@book{Szyperski2002,

author = {Szyperski, Clemens},

title = {Component Software: Beyond Object-Oriented Programming},

year = {2002},

isbn = {0201745720},

edition = {2nd},

publisher = {Addison-Wesley Longman Publishing Co., Inc.},

address = {Boston, MA, USA},

}

@book{deitel2010,

title={Java: como programar},

author={Deitel, H.M. and Deitel, P.D.},

isbn={9788576055631},

edition={8 ed.},

url={http://books.google.com.br/books?id=xWMVRAAACAAJ},

year={2010},

publisher={PEARSON BRASIL},

address={São Paulo, SP, Brasil}

}

@book{pressman2011,

title={Engenharia de Software: Uma Abordagem Profissional},

author={Roger S. Pressman},

edition={7 ed.},

isbn={9788563308337},

url={http://books.google.com.br/books?id=eRIOuQAACAAJ},

year={2011},

publisher={AMGH},

address={Porto Alegre, RS, Brasil}

}

@book{coulouris2007,

title={Sistemas Distribu{\'\i}dos: Conceitos e Projeto},

author={COULOURIS, G. and DOLLIMORE, J. and KINDBERG, TIM},

isbn={9788560031498},

url={http://books.google.com.br/books?id=KSZ1rIRWmUoC},

year={2007},

edition{4 ed.},

publisher={BOOKMAN COMPANHIA ED},

address={Porto Alegre, RS, Brasil}

}

@phdthesis{rocha2008,

address={Campinas, SP, Brasil},

author={Tarcisio da Rocha},

school={Instituto de Computa\c{c}\~ao, Universidade Estadual de Campinas},

title={{Servi\c{c}os de transa\c{c}\~ao abertos para ambientes din\^amicos}},

year={2008}

}

@inproceedings{clarke2001,

author = {Clarke, Michael and Blair, Gordon S. and Coulson, Geoff and Parlavantzas, Nikos},

title = {An Efficient Component Model for the Construction of Adaptive Middleware},

booktitle = {Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg},

series = {Middleware '01},

year = {2001},

isbn = {3-540-42800-3},

pages = {160--178},

numpages = {19},

url = {http://dl.acm.org/citation.cfm?id=646591.697779},

acmid = {697779},

publisher = {Springer-Verlag},

address = {London, UK, UK},

}

@masterthesis{nascimento2013,

address={Aracaju, SE, Brasil},

author={Sidney Cassemiro do Nascimento},

school={Universidade Federal de Sergipe},

title={{Um Framework Extensível para Interoperabilidade Dinâmica entre Componentes Distribuídos}},

year={2013}

}