Uma Ferramenta para Configuração e Implantação de Sistemas Distribuídos de Tempo-Real Baseados em Componentes

Sandro S. Andrade, Aristoteles M. Silva e Cleber N. Ramos

Faculdade Ruy Barbosa
Rua Theodomiro Batista, 422 - Rio Vermelho - Cep: 41.940-320
Fone/Fax : +55 (71) 3205-1733 - Salvador - Bahia - Brasil

{sandros,clebernr,aristotelesms}@frb.br

Abstract. *Meeting demands related to distribution, flexibility, reusability, and interoperability in real-time systems has been the goal of many current researches, some of them devoted to the use of distributed component technology as an important mechanism for complexity management and temporal predictability. This paper presents the design and implementation of a tool aiming the visual configuration and deployment of component-based distributed systems built atop of CIAO: a recent implementation of the CORBA Component Model (CCM) standard, with real-time extensions. The proposed tool provides visual mechanisms for implementing and importing CIAO components, interconnecting components into an assembly, configuring of real-time parameters, and deploying of the component-based system in different network nodes.*

Resumo. *O atendimento de demandas relacionadas a distribuição, flexibilidade, reutilização e interoperabilidade em sistemas de tempo-real tem sido o foco de diversas pesquisas recentes, algumas direcionadas ao uso de componentes distribuídos como tecnologia para gerenciamento da complexidade e para previsibilidade temporal. Este artigo apresenta o projeto e implementação de uma ferramenta para configuração e implantação visual de sistemas baseados no CIAO: uma implementação recente do CORBA Component Model (CCM), com extensões para tempo-real. A ferramenta disponibiliza mecanismos gráficos para implementação e importação de componentes CIAO, combinação visual entre componentes, configuração de requisitos temporais e implantação remota da montagem em diferentes nós de uma rede.*

1. Introdução

Nos últimos anos, a crescente evolução das tecnologias de processamento e comunicação de dados tem demandado novos requisitos para sistemas computacionais. A utilização de plataformas flexíveis, escaláveis, reutilizáveis e interoperáveis são características atualmente desejadas e a complexidade gerada pela introdução de tais requisitos requer a adoção de metodologias, mecanismos e ferramentas que auxiliem o desenvolvimento controlado e produtivo de tais sistemas.

A utilização de componentes distribuídos de software [Gimenes and Huzita 2005, Heineman and Councill 2001] tem sido uma prática promissora para a obtenção de aplicações e plataformas flexíveis, reutilizáveis e escaláveis. Ao possibilitar a construção

de sistemas através da combinação facilitada de peças de software, a tecnologia de componentes disponibiliza uma infra-estrutura adequada ao desenvolvimento produtivo de plataformas, tais como os *frameworks* de aplicação, e de aplicações em diversos domínios.

Em particular, os sistemas distribuídos de tempo-real têm sido o foco de diversas pesquisas interessadas no estudo da viabilidade de utilização, em tais sistemas, de soluções de *middleware* para objetos e componentes distribuídos. Tecnologias tais como o TAO (*The ACE ORB*) [Schmidt et al. 1998] e o CIAO (*Component-Integrated ACE ORB*) [N. Wang and Subramonian 2004] representam esforços importantes no sentido de adequar as tecnologias atuais da engenharia de software para uso em sistemas de tempo-real.

Em adição à inerente complexidade introduzida pelas demandas atuais, implementações de determinados modelos de componentes distribuídos tendem a inserir artifícios relacionados diretamente aos serviços e funcionalidades disponibilizados pelo *middleware*, tais como arquivos de definição de interfaces, métodos de *callback* e descritores XML de configuração e implantação. Ao mesmo tempo em que constituem ferramentas indispensáveis para o desenvolvimento de sistemas distribuídos modernos, as soluções de *middleware* para componentes ainda requer, do desenvolvedor, um conhecimento especializado sobre o modelo implementado e sobre a arquitetura utilizada.

Este artigo apresenta o projeto e implementação do ATOME: uma ferramenta para configuração e implantação de sistemas distribuídos de tempo-real baseados em componentes. O ATOME possibilita o desenvolvimento produtivo de sistemas distribuídos de tempo-real baseados no *CORBA Component Model*[1] (CCM) [OMG 2001], através da disponibilização de mecanismos para: geração automática dos artefatos requeridos pelo CIAO para criação e implantação de componentes, conexão e configuração visual de componentes, criação do mapa de nós do ambiente de implantação, configuração dos atributos temporais de instâncias e implantação distribuída da montagem.

Dentre os objetivos do ATOME destacam-se: i) proporcionar uma transparência ao desenvolvedor em relação aos artifícios requeridos pelo CCM/CIAO, possibilitando o foco no negócio da aplicação, ii) possibilitar a conexão e configuração visual dos componentes de uma montagem, incluindo restrições temporais e iii) automatizar o processo de implantação distribuída do sistema.

O restante do artigo está organizado como segue. A seção 2 apresenta os fundamentos e tecnologias que motivaram a construção da ferramenta proposta. A seção 3 apresenta a arquitetura e o projeto do ATOME, suas funcionalidades e aspectos relacionados à sua implementação, enquanto a seção 4 apresenta os trabalhos correlatos a este projeto. Finalmente, a seção 5 apresenta as conclusões e trabalhos futuros.

2. Componentes Distribuídos para Tempo-Real

A aplicação de tecnologias da engenharia de software em sistemas de tempo-real tem sido o foco de pesquisas recentes [Andrade and Macêdo 2007, A. Tesanovic and Norstom 2004]. Em particular, soluções para componentes distribuídos de tempo-real [Hatcliff et al. , N. Wang and Subramonian 2004] têm sido experimentadas em cenários tais como sistemas industriais de supervisão e controle, sistemas robóticos

[1]Modelo de componentes implementado pelo CIAO. Padronização para componentes distribuídos criada pelo Object Management Group (OMG), em 2001.

e sistemas embarcados. Esta seção apresenta brevemente o *CORBA Component Model* (CCM) e as extensões para tempo-real implementadas pelo *Component-Integrated ACE ORB* (CIAO).

2.1. CORBA Component Model (CCM)

O *Object Management Group* (OMG) disponibilizou em junho de 2001 a versão 3 do CORBA, contemplando um modelo e uma especificação de componentes denominada *CORBA Component Model* (CCM) [OMG 2001]. O CCM define um modelo de componentes distribuídos, mecanismos para conexão entre componentes e padronizações para implantação, configuração e manutenção de componentes.

Um componente CCM é uma unidade de implantação e conexão que implementa funcionalidades reutilizáveis do sistema computacional em questão. Essas funcionalidades são disponibilizadas através de métodos pertencentes à(s) interface(s) suportada(s) pelo componente. Uma interface suportada é uma interface CORBA da qual o componente realiza uma implementação, definindo os métodos que podem ser invocados por aplicações clientes daquele componente.

As conexões entre componentes são realizadas através de mecanismos denominados *ports*. O CCM define quatro tipos de *ports*: i) facetas (*facets*) - representam uma funcionalidade provida pelo componente, ii) receptáculos (*receptacles*) - representam uma funcionalidade requerida pelo componente e, em conjunto com as facetas, constituem um mecanismo para conexão síncrona (bloqueante) de componentes, iii) produtor de eventos (*event sources*) - representam geradores de eventos e iv) consumidores de eventos (*event sinks*) - representam receptores de eventos e, em conjunto com os produtores de eventos, constituem um mecanismo para conexão assíncrona (não-bloqueante) de componentes.

Todas as operações de conexão de facetas e receptáculos, bem como de produtores e depósitos de eventos, são realizadas pelo *middleware*, a partir de indicações nos arquivos XML descritores de implantação. Além dos *ports*, o CCM define pontos de configuração do componente denominados atributos. O valor do atributo é informado, via arquivo descritor, no momento da implantação do componente. Um atributo serve para adequar o funcionamento do componente a uma situação particular.

2.2. Component-Integrated ACE ORB (CIAO) e Tempo-Real

O CIAO (*Component-Integrated ACE ORB*) [N. Wang and Subramonian 2004] é uma implementação da especificação CCM desenvolvida pelo DOC Group[2] e baseia-se na utilização de duas tecnologias bastante consolidadas na área de *middleware* para tempo-real: o ACE (*ADAPTIVE Communication Environment*) [Schmidt and Huston 2002] - um *framework* para desenvolvimento de sistemas distribuídos de alto desempenho - e o TAO (*The ACE ORB*) [Schmidt et al. 1998] - uma implementação da especificação CORBA 2.x (baseada em objetos distribuídos), com extensões para tempo-real.

Enquanto o CCM destaca-se dos demais modelos devido aos mecanismos expressivos para conexão e configuração de componentes e serviços para implantação disponibilizados, o CIAO sobressai-se em relação às outras implementações do CCM em função das seguintes características: i) implementação de alto desempenho (devido às

[2]http://www.cs.wustl.edu/ schmidt/TAO.html

otimizações realizadas em todos os participantes da arquitetura CORBA), ii) execução previsível de componentes (através da configuração, via XML, dos parâmetros definidos pelo RT-CORBA [(OMG) 2003] e da implementação do *container* de tempo-real), iii) serviço para configuração e implantação (todo o processo de implantação e conexão de componentes em diferentes nós da rede, bem como a gerência do repositório de componentes, são realizados pelo DAnCE - *Deployment And Configuration Engine*) e iv) mecanismos para reconfiguração dinâmica (adição e remoção de instâncias e conexões são realizadas pelo ReDaC - *Redeployment and Reconfiguration* - sem requerer a interrupção do sistema).

O RT-CORBA [(OMG) 2003] é uma especificação, baseada no CORBA 2.x, que define extensões para a implementação de objetos distribuídos para sistemas de tempo-real com prioridades fixas. As invocações com prioridades no RT-CORBA seguem um dos modelos especificados: *client-propagated* e *server-declared*. No modelo *client-propagated*, a prioridade é informada pelo cliente no momento da invocação e o servidor ajusta o ambiente para executar a requisição. No modelo *server-declared* as prioridades são pré-definidas pelo servidor no momento da implantação. Ao adquirir uma referência para um componente, o cliente obtém a prioridade declarada pelo servidor e esta não poderá ser alterada.

Com o objetivo de garantir a previsibilidade das invocações, o RT-CORBA define dois modelos de reserva de recursos: o *thread-pool* e o *thread-pool with lanes*. No modelo *thread-pool*, um conjunto de *threads* (*pool*) é previamente criado e os seguintes parâmetros podem ser configurados: número de *threads* estáticas, número de *threads* dinâmicas, prioridade das *threads* e tamanho do *buffer* de requisições. O RT-CORBA gerencia e manipula o *pool* de *threads* com base nessa parametrização, com o objetivo de prover um ambiente de execução previsível. No modelo *thread-pool with lanes*, diversos grupos (*lanes*) de *threads* podem ser criados dentro de um mesmo *pool*. Cada grupo possui seus próprios parâmetros de configuração e pode tomar *threads* "emprestadas" de outros grupos, em situações de escassez de recursos.

O CIAO disponibiliza um mecanismo facilitado para configuração dos parâmetros do RT-CORBA através da utilização de descritores XML, em contraste à forma custosa e propensa a erros do CORBA 2.x, realizada via código-fonte.

3. A Ferramenta Proposta

A ferramenta proposta neste trabalho, ATOME, integra as tecnologias oferecidas pelo CIAO e disponibiliza um ambiente fácil e produtivo para o desenvolvimento de sistemas baseados em componentes CCM. A Figura 1 apresenta a arquitetura do ATOME e o seu relacionamento com as tecnologias do CIAO.

O ATOME possui duas macro-funcionalidades importantes: geração de versões iniciais de componentes CCM e configuração e implantação de montagens de tempo-real. Motivado pela considerável complexidade de criação de um componente CIAO (principalmente por desenvolvedores inexperientes), o ATOME disponibiliza recursos gráficos para criação de componentes através da indicação da(s) interface(s) suportada(s), facetas, receptáculos, produtores, consumidores e atributos que o compõem. Para a definição de uma montagem, o ATOME disponibiliza um editor gráfico para conexão e configuração de atributos e parâmetros temporais dos componentes gerados pela ferramenta ou impor-

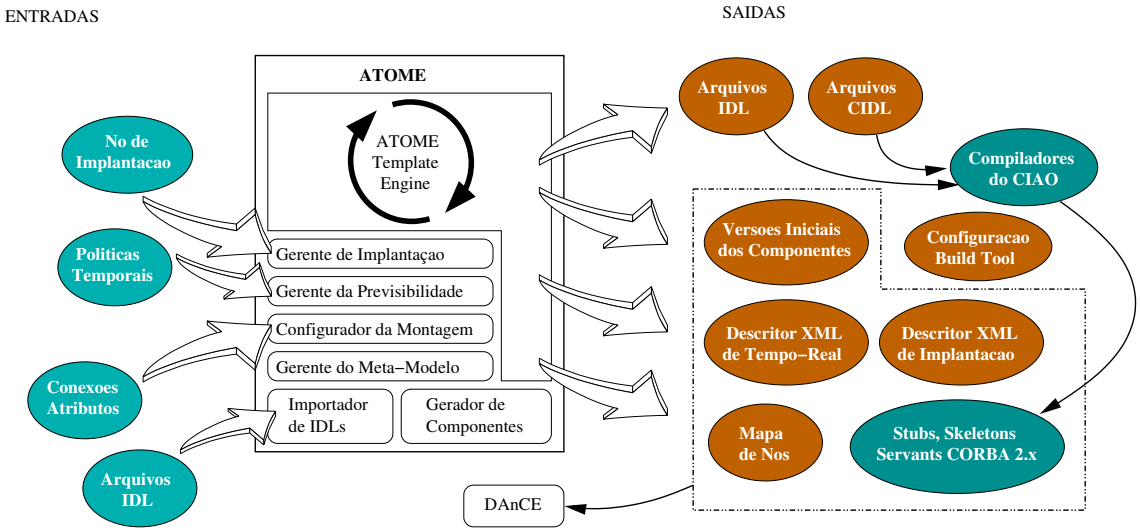

Figura 1. Arquitetura do ATOME e tecnologias auxiliares.

tados através de seus arquivos IDL.

Os principais participantes definidos na arquitetura do ATOME são:

- Gerador de Componentes: responsável pela criação de componentes CCM/CIAO através de indicações visuais realizadas na ferramenta, tais como criação de interfaces, atribuição de interfaces a facetas, criação de receptáculos, criação de atributos, definição de tipos de eventos e criação de produtores e consumidores. Interage com o *ATOME Template Engine*, responsável pela geração dos seguintes artefatos: arquivos IDL e CIDL, versão inicial (obviamente sem lógica de negócio) da implementação do componente e arquivo de configuração da ferramenta de compilação MPC (*Make Project Creator*);
- Importador de IDLs: componentes CCM/CIAO previamente criados podem ser importados para uso em futuras montagens. O importador de IDL realiza o *parsing* do arquivo IDL, informando a estrutura do componente para o Gerente do Meta-Modelo;
- Gerente do Meta-Modelo: responsável pelo armazenamento de meta-informações referentes aos componentes importados/criados e à montagem sendo configurada. Implementa verificações de consistência na montagem, tais como compatibilidade entre *ports* (impede conexões inapropriadas ou entre tipos distintos) e detecção de componentes isolados;
- Configurador da Montagem: interage com o Gerente do Meta-Modelo a partir de requisições de conexão e configuração de componentes realizadas pelo usuário. Armazena o mapa de nós do domínio de implantação, contendo os endereços IP e portas das máquinas que podem hospedar componentes;
- Gerente da Previsibilidade: disponibiliza mecanismos para criação das políticas do RT-CORBA (*thread pools* e modelos de invocação com prioridade). Possibilita a atribuição dessas políticas às instâncias criadas na montagem;
- Gerente de Implantação: responsável pela coleta dos artefatos necessários à implantação distribuída da montagem. Utiliza os serviços subjacentes, disponibilizados pelo DAnCE;

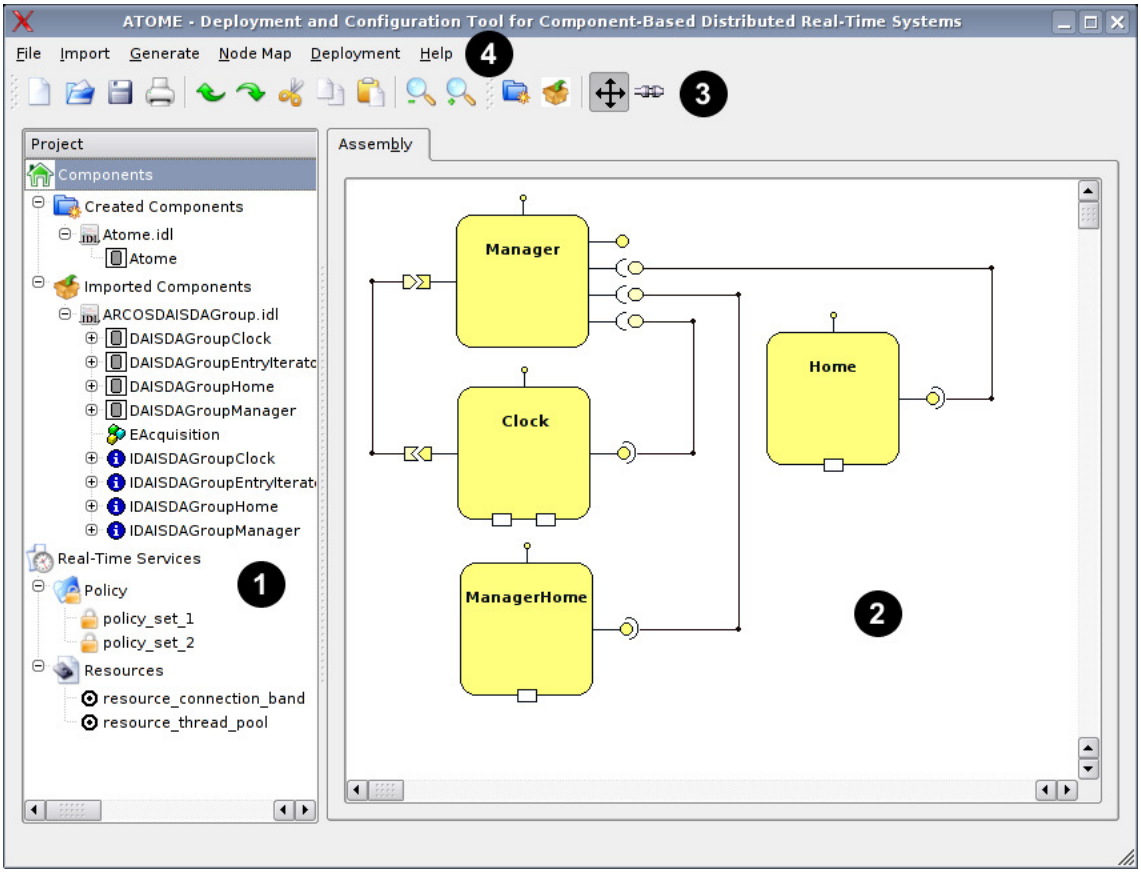

Figura 2. A ferramenta ATOME.

- *ATOME Template Engine*: núcleo central para geração de artefatos. Utiliza arquivos *template* contendo versões parametrizadas de todos os artefatos gerados.

A Figura 2 apresenta a ferramenta ATOME exibindo a construção de uma montagem. A área 1 apresenta a visão dos recursos disponíveis para montagem. A árvore "*Components*" apresenta os componentes criados pelo usuário através da ferramenta, bem como os componentes importados através dos seus arquivos IDL. A árvore "*Real-Time Services*" contém os recursos e políticas de tempo-real definidos pelo usuário (*thread-pools*, *lanes* e recursos relacionados à rede de comunicação). Componentes e serviços de tempo-real podem ser facilmente criados através de menus de contexto (botão direito do *mouse*) disponíveis em ramos das árvores apresentadas.

A área 2 apresenta o editor de montagens implementado no ATOME. Nesse editor o usuário pode solicitar a criação de uma nova instância de componente, indicar o nó do domínio no qual a instância será implantada, realizar conexões com outras instâncias criadas, ajustar valores de atributos e atribuir políticas de tempo-real. Com essa abstração, usuários iniciantes na tecnologia CCM/CIAO poderão focar nas questões de negócio da aplicação, deixando para o ATOME a responsabilidade da geração correta dos artefatos necessários. Ressalta-se a alta complexidade e propensão a erros da construção manual dos descritores XML de implantação e de tempo-real. As áreas 3 e 4 apresentam, respectivamente, a barra de tarefas e as operações de menu da ferramenta ATOME.

	Cadena	CoSMIC	ATOME
Plataforma de *middleware* utilizada	OpenCCM e CIAO	ACE, TAO e CIAO	CIAO
Suporte a tempo-real	Sim	Sim	Sim
Mecanismo para montagem e criação de componentes	Edição de arquivos	UML	Gráfico
Integração com outra ferramenta	Eclipse	Não	Não
Suporte ao processo de implantação	Não	Sim	Sim

Tabela 1. Comparação do ATOME com as demais ferramentas.

3.1. Aspectos de Implementação

O ATOME foi projetado e desenvolvido no sistema operacional GNU/Linux, utilizando a linguagem Standard C++ e as tecnologias TrollTech Qt3 e KDevelop. O Qt3 é um *toolkit* para desenvolvimento de aplicações visuais modernas, utilizando a linguagem C++. Caracterizado pela sua excelente portabilidade entre os ambientes GNU/Linux, Microsoft Windows e MAC OS, o Qt3 disponibiliza um conjunto extensivo de classes para *multithreading*, manipulação de XML, multimídia, banco de dados, além dos convencionais recursos para construção de interfaces gráficas.

O KDevelop é um IDE (*Integrated Development Environment*) para desenvolvimento de software livre na plataforma GNU/Linux. O KDevelop atualmente suporta cerca de doze linguagens de programação e possui forte integração com as ferramentas do Qt, constituindo um ambiente produtivo para o desenvolvimento de sistemas modernos. Todos os módulos do ATOME foram integralmente desenvolvidos a partir das API's disponibilizadas pelo Qt facilitando, dessa forma, a sua migração para outras plataformas. O ATOME encontra-se atualmente em processo de migração para o Qt4, o que possibilitará a sua execução no ambiente Microsoft Windows sem a necessidade de adoção da licença proprietária requerida pelo Qt3 neste ambiente.

4. Trabalhos Correlatos

Dentre as atuais ferramentas de auxílio ao desenvolvimento de sistemas de tempo-real baseados em componentes pode-se destacar: o CoSMIC (*Component Synthesis using Model Integrated Computing*) [Gokhale et al. 2003] e o Cadena [Hatcliff et al.]. O CoSMIC reúne uma série de aplicações para especificação, verificação, implementação, configuração e montagem de sistemas de tempo-real distribuídos baseados em componentes e na tecnologia MDA (*Model-Driven Architecture*). O CoSMIC tem como objetivo a geração dos artefatos de um sistema de tempo-real a partir de um modelo UML fornecido pelo desenvolvedor.

O Cadena [Hatcliff et al.] promove extensões para tempo-real a partir de uma implementação do CCM denominada OpenCCM. Além das extensões para tempo-real, o Cadena disponibiliza uma ferramenta para composição de montagens e geração dos descritores XML utilizados na implantação. A tabela 1 apresenta a comparação entre o ATOME e as demais ferramentas.

5. Conclusões e Trabalhos Futuros

Este artigo apresentou o ATOME: uma ferramenta para configuração e implantação visual de sistemas distribuídos de tempo-real baseados em componentes CCM/CIAO. Com a adoção e amadurecimento prático do desenvolvimento de sistemas baseados em componentes, ferramentas de configuração e implantação de montagens se tornam fundamentais para o desenvolvimento produtivo de sistemas. Neste contexto, o papel do montador de software é indispensável para a implantação de metodologias e práticas para o desenvolvimento ágil de sistemas.

Dentre os trabalhos futuros da ferramenta destacam-se: projeto e implementação do módulo de integração com repositórios do DAnCE, suporte a "*profiles*"com o objetivo de direcionar montagens de sistemas construídos a partir de *frameworks* baseados em componentes, migração para o Qt4 (disponibilizando uma versão *open source* para o ambiente Microsoft Windows) e implementação do módulo de gerência do ambiente distribuído, responsável pelo monitoramento das instâncias em cada nó do domínio. Maiores informações sobre a ferramenta ATOME, bem como obtenção de versões binárias e de código-fonte podem ser encontradas em http://atome.sourceforge.net.

Referências

A. Tesanovic, D. Nystrom, J. H. and Norstom, C. (2004). Aspects and components in real-time system development: Towards reconfigurable and reusable software. *Journal of Embedded Computing*.

Andrade, S. and Macêdo, R. (2007). Engineering components for flexible and interoperable real-time distributed supervision and control systems. In *12th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'2007). September 25-28, 2007. Patras-Greece*.

Gimenes, I. M. D. S. and Huzita, E. H. M. (2005). Desenvolvimento baseado em componentes: Conceitos e técnicas. *Ciencia Moderna*.

Gokhale, A., Schmidt, D., Lu, T., Natarajan, B., and Wang, N. (2003). Cosmic: An MDA generative tool for distributed realtime and embedded applications.

Hatcliff, J., Deng, W., Dwyer, M., Jung, G., and Ranganath, V. Cadena: An integrated development, analysis, and verification environment for component-based systems.

Heineman, G. T. and Councill, W. T. (2001). Component based software engineering: Putting the pieces together. *Addison-Wesley Professional; 1st edition*.

N. Wang, C. Gill, D. C. S. and Subramonian, V. (2004). Configuring real-time aspects in component middleware. *International Symposium on Distributed Objects and Applications - DOA 2004*.

OMG, O. M. G. (2001). *CORBA Component Model*. http://www.omg.org/technology/documents/formal/components.htm.

(OMG), O. M. G. (2003). *Real-Time CORBA*. http://www.omg.org/technology/documents/formal/RT_dynamic.htm.

Schmidt, D. C. and Huston, S. D. (2002). C++ Network Programming: Mastering complexity using ACE and patterns. *Addison-Wesley Longman*.

Schmidt, D. C., Levine, D. L., and Mungee, S. (1998). The design of the TAO real-time object request broker. *Computer Communications*, 21(4).