

DanCE : A QoS-Enabled Component Deployment and Configuration Engine

Autores originais:

Gan Deng, Jaiganesh Balasubramanian, William Otte,
Douglas Schmidt e Aniruddha Gokhale

<http://www.dre.vanderbilt.edu/cosmic/html/launching.html>

Apresentador:

Amadeu Andrade Barbosa Júnior <ajunior@inf.puc-rio.br>

INF2556 – Seminários de Sistemas Distribuídos
Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro

Agenda

A yellow decorative shape that starts as a thin line on the left and expands into a wide, upward-pointing triangle on the right, positioned behind the title.

- Motivação
- Desafios gerais
- Abordagem do DanCE
- Resolvendo os desafios
- Trabalhos futuros
- Trabalhos relacionados
- Algumas observações sobre CCM
- Comentários

Contexto

- Aplicações DRE¹ : *Distributed and Real-Time Embedded Systems*
- Middlewares para componentes podem tornar o software mais flexível
- Separação entre funcionalidades da aplicação e atividades do ciclo de vida
- Entende-se como atividades do ciclo de vida: configuração e implantação

¹ *chamaremos daqui para frente apenas de sistemas de tempo real*

Motivação

- J2EE e .NET não se configuram como soluções pois não suportam QoS em suas arquiteturas
- Middlewares de componentes com suporte a QoS devem:
 - Implantar componentes compostos (*assemblies*)
 - Ativar e desativar componentes automaticamente
 - Inicializar e configurar os servidores de componentes
 - Garantir critérios de tempo real
- Devem simplificar a configuração, implantação e gerenciamento dos serviços
- Falta de mecanismos portáveis, reusáveis e padronizados dificulta a adoção de middlewares de componentes

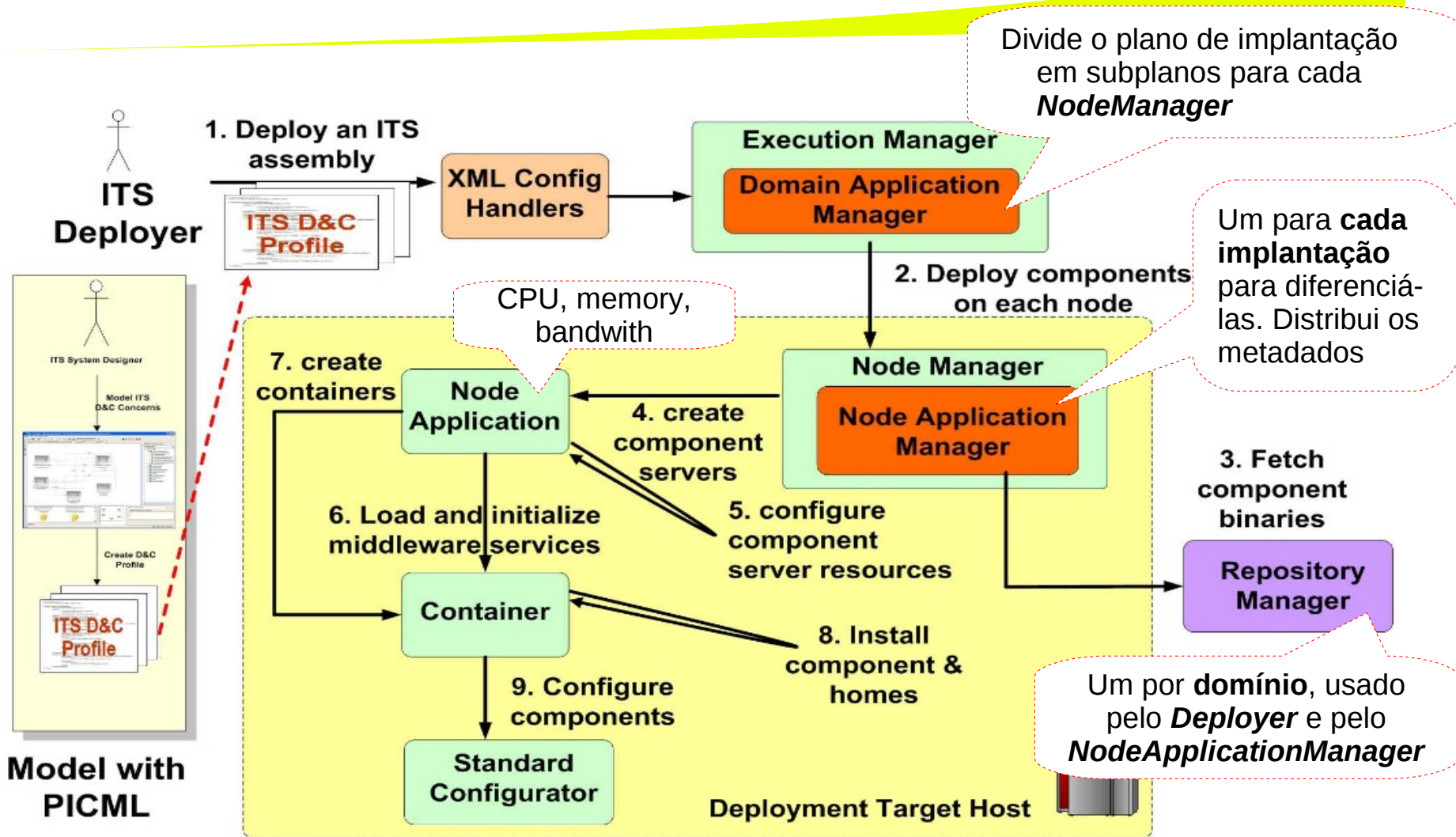
Desafios gerais

1. Métodos eficientes de armazenar e obter implementações de componentes
 - Diferentes linguagens e plataformas, respeitando restrições de tempo
2. Prover tarefas de ativação, suspensão e desativação dos componentes
 - Recursos compartilhados
 - Suspende caso ocioso ou desativar se possível
3. Configurar QoS no servidor de componentes
 - Alterar políticas de QoS para servidores, contêineres e componentes
4. Configurar e implantar serviços comuns do middleware
 - Middleware precisa manter compatibilidade com serviços CORBA 2.x como Naming, Trader, Publish/Subscriber

Abordagem do DAnCE

- Compatível com a especificação de *Deployment & Configuration* (D&C) da OMG, que é uma extensão para implantação do CCM, que substitue a anterior P&D
- D&C padroniza o uso de XML para armazenar e trocar metadados e a adoção de um ambiente de execução
- DAnCE é construído a partir do CIAO (implementação CCM do TAO)
- A infra-estrutura do DAnCE é implementada como objetos CORBA 2.x
- Assume-se o uso de ferramentas externas de *MDD* (*Model-driven Development*) para geração dos XML padronizados

Visão Geral do DAnCE



Deploy and Configure with DAnCE

DAnCE: Resolvendo os desafios (1)

- *RepositoryManager*
 - Permite obtenção/submissão das implementações versionadas
 - Pode buscá-las em outros locais (URL no plano de implantação)
 - Executa colocado no *host* do *NodeApplicationManager*
 - Instala¹ a implementação para os contêiners
 - *NodeApplicationManager* pode fazer cache local para diminuir o custo de re-implantação de todos componentes

¹ *considera-se que publicar o componente no repositório implica em manter instalado (sistema de arquivos) para carga via contêiners*

DAnCE: Resolvendo os desafios (2)

- *DomainApplicationManager*
 - Coordena o ciclo de vida dos *assemblies* de componentes
 - Mantém estados PREATIVO, ATIVO, PASSIVO, INATIVO sobre cada componente
 - Garante **só** conectar e ativar os componentes **quando** todos outros do *assembly* estiverem pré-ativados
 - Garante (pela tabela de despacho do POA e interceptadores do ORB) que as invocações remotas **se concretizem antes** de passar ao estado passivo/inativo

DAnCE: Resolvendo os desafios (3)

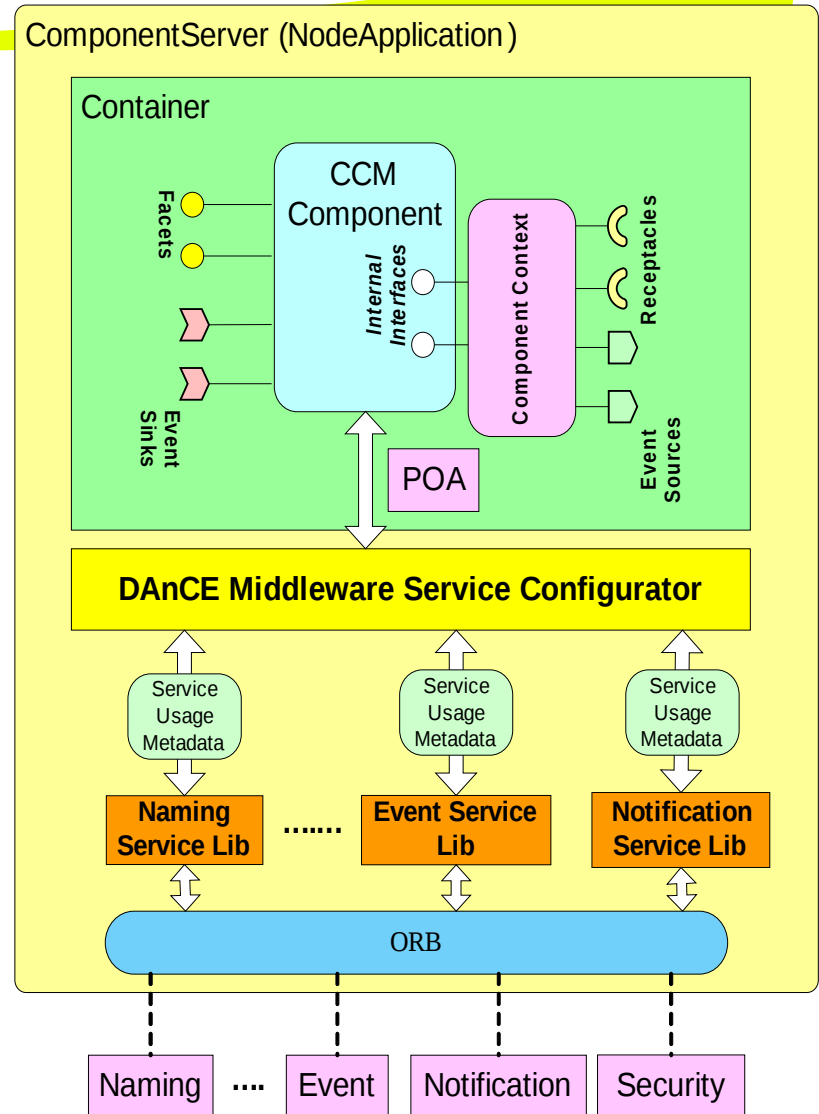
- Configurador de QoS dos servidores de componentes
 - Extensão às configurações do servidor *NodeApplication*
 - Argumentos do console do ORB e parâmetros dos serviços do ORB
 - OBS: TAO usa fábricas diferenciadas para:
 - modelos de concorrência
 - filas de prioridades de conexões e despacho
 - base para diversos serviços RT (ex: *RTEventService*)
 - Abstrações comuns aos modelos de *servants*

```
<rtpolicyset id="HIGH_PRIORITY">
  <priority_model_policy
    type="server_declared"
    priority="6"/>
</rtpolicyset>

<instance id="myInstanceBasicSensor">
  <name>BasicSensorId</name>
  <implementation>
    BasicSensorImpl
  </implementation>
  <configProperty>
    <extension class="RT-POLICY-SET"
      origin="CIAO"> HIGH_PRIORITY
    </extension>
  </configProperty>
</instance>
```

DAnCE: Resolvendo os desafios (4)

- Configurar serviços gerais do middleware durante implantação
 - Implementa um *ServiceConfigurator* colocado em cada *NodeApplication*
 - Usa as interfaces padrão de cada serviço
 - Os serviços TAO possuem fachadas compatíveis com o CIAO
 - Uso declarativo via descritores XML



Trabalhos relacionados

- **OpenCCM** – ainda usa *Package and Deployment* do CCM
- **ADAGE** (?) – na verdade os autores citam um trabalho do grupo que, posteriormente, criou o ADAGE, o artigo fala sobre implantação do CCM via Globus Toolkit.
Simplesmente é dito: *nosso foco é em sistemas de tempo real e não em grades.*
- **ProActive** – justificativa clássica de ser uma ferramenta para Java (no caso objetos ativos). ... dito que DAnCE “*vai além*” pois especifica **interdependências entre componentes** e **garante a consistência do sistema em tempo de implantação.**

Trabalhos futuros

- Integrar mecanismos de multicast confiável do TAO aos gerentes de *Node*, *Application* e *Domain*
- Suportar reconfiguração dinâmica dos *assemblies*, re-implantação e migração
- Prover sincronização de estados e re-implantação/recuperação para melhorar tolerância a falhas
- Aplicar técnicas de verificação e especialização, a partir dos metadados, para otimizar o sistema de tempo real
 - A lógica da composição e o plano de implantação estão completamente descritos nos metadados em XML

Alguns observações sobre CCM

- Definição de *assemblies*:
 - *“In order to instantiate, or deploy, a component-based application, instances of each subcomponent must first be created, then interconnected and configured.*
 - (...) subcomponents might be distributed among a set of independent, interconnected nodes called domain.”*
- Diferente dos *composites*† do Fractal
- A especificação de P&D (Package & Deployment) do CCM é precária e confusa. Essa nova especificação de D&C é mais simples.
- XML não é *human-friendly*, assim assume-se o uso ferramentas gráficas de *MDD*

Comentários / Dúvidas

- Maior contribuição: implementação do OMG-D&C e indicar sua adequação a um cenário de uso prático
- Não fica claro se já existiam outras implementações do OMG-D&C (provavelmente não)
- Não fica claro se implementou todo OMG-D&C. Não comentam sobre *Target Management Model* e recursos
- Conclusões há “... *DAnCE (...) targeted for deploying and configuring DRE systems based on Lightweight CORBA Component Model (CCM)*”
 - Como o DAnCE trata com componentes CCM não-lightweight ? Qual o impacto entre ser ou não CCM lightweight ?
- Fraco na comparação com trabalhos relacionados

Bibliografia

@inproceedings{dance,

author = {Gan Deng and Jaiganesh Balasubramanian and William Otte
and Douglas C. Schmidt and Aniruddha Gokhale},

title = {DAnCE: A QoS-enabled Component Deployment and
Configuration Engine},

booktitle = {3rd Working Conference on Component Deployment},

year = {2005},

pages = {67--82},

url = {<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.60.36>},

}

Estudo de Caso: *Inventory Tracking System*

Gerenciar o armazenamento e movimentação de bens com restrições de tempo e confiabilidade

- Gerenciamento do entreposto – tomada de decisão sobre as localizações e destinos
- Controle do fluxo de materiais – trata os detalhes como cálculo das rotas e reservas nos meios de transporte para garantir o envio até o destino
- Camada física – manipula os sensores, guindastes,...