MAC-5759 – Sistemas de Objetos Distribuídos

Quality Objects – QuO

Germano Capistrano Bezerra

Junho de 2002

1

Roteiro

- Apresentação
- *CORBA* e Aplicações Distribuídas
- *QoS* em Ambientes Distribuídos
- Visão Geral de *QuO*
- *QDL – Quality Description Languages*
- Componentes Internos de *QuO*
- Cenário de Desenvolvimento
- Conclusões

2

Objetivos do Seminário

- Apresentar uma visão geral de *Quality Objects* visando discutir conceitos complementares ao curso de SOD, fomentando o interesse de futuros estudos no assunto.

3

CORBA e Aplicações Distribuídas

- Aplicações distribuídas estão cada vez mais difundidas
- *CORBA* contribuiu bastante para essa difusão:
  - Interface funcional é especificada de forma separada da implementação em si
  - Interoperabilidade mesmo não considerando detalhes de implementação
  - Transparência de SO, HW, localização física, linguagem de implementação

4

CORBA e Aplicações Distribuídas

- Na prática, *CORBA* funciona bem:
  - Qdo. os recursos são plenos
  - As condições de ambiente são estáveis
  - O ambiente de execução é previsível
- Isso vale para *LAN´s*
- Em *WAN´s*:
  - Muitos recursos são ou podem tornar-se escassos
  - Variáveis de sistema são imprevisíveis
  - Há uma enorme combinação de cenários possíveis

5

CORBA e Aplicações Distribuídas

- Problemas encontrados em muitas aplicações:
  - Desconsideração das condições variáveis de ambientes distribuídos
  - Quando se considera, falta experiência para tratar condições diferentes das condições locais às quais o programador está acostumado
  - *IDL* esconde aspectos de implementação que lidam com *tradeoffs* que um objeto poderá fazer
  - Não há padrão de reusabilidade de componentes que lidam com esses aspectos

6

QoS em Ambientes Distribuídos

- *QoS* – Aplicações distribuídas constantemente possuem requisitos de Qualidade de Serviço em paralelo aos requisitos funcionais especificados na *IDL* de *CORBA*
  - Desempenho em tempo real
  - Sincronia de dados
  - Segurança
  - Dependência
  - Tolerância a falhas

7

QoS em Ambientes Distribuídos

- Aplicações podem prover *QoS*, com o custo de se tornarem muito específicas
- *IDL* e *ORB´s*
  - Especificam, monitoram e controlam os aspectos funcionais da aplicação
- *QuO* – Arquitetura proposta pela *BBN Technologies* para implementar *QoS* em objetos *CORBA*
  - *QuO* especifica, monitora e controla *QoS*

8

QoS em Ambientes Distribuídos

- *QuO*
  - Estados de *QoS*, definidos em "contratos" entre os clientes e os provedores do serviço
  - Elementos de sistema monitorados e controlados para medir e controlar os estados de *QoS*
  - Comportamento adaptativo para as mudanças ocorridas nos estados
- Com *QuO*, tem-se o controle tanto de aspectos funcionais como de estratégias de implementação a serem adotadas

9

QoS em ambientes Distribuídos

- Os objetivos de *QuO* são:
  - Aplicações distribuídas mais robustas – seleção do melhor mecanismo específico em vez de um mecanismo geral
  - Controle sobre como as aplicações distribuídas podem ser adaptativas – possuir a informação correta para a decisão de como o sistema deve se comportar
  - Prover reusabilidades de mecanismos adaptativos – *Quality Description Languages*

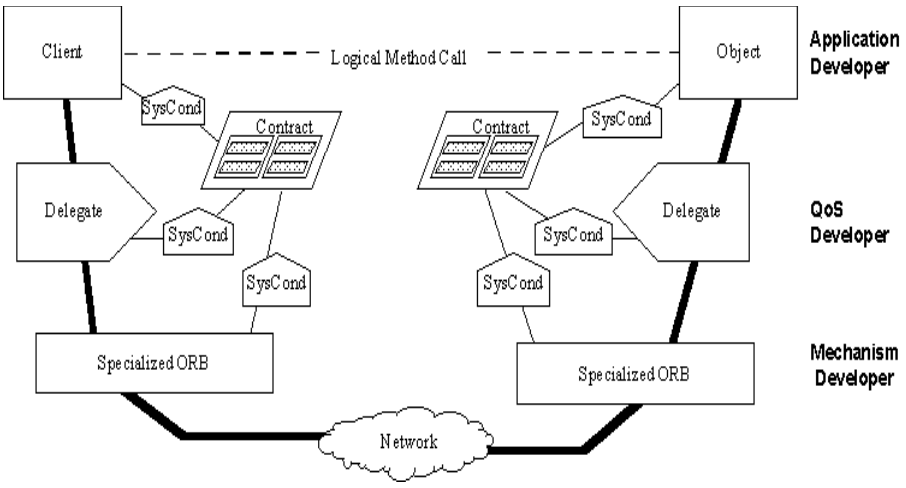
10

Visão Geral de *QuO*

- *QuO* especifica os aspectos de *QoS* de forma análoga à especificalção funcional da *IDL*
- O fluxo de requisição em QuO é modificado em relação ao fluxo de *CORBA*
- Surgem alguns novos elementos:
  - Delegados (*Delegates)*
  - "Contratos" (*Contracts*)
  - Objetos de condições de sistema (*System condition objects*)

11

Visão Geral de *QuO*



Invocação remota de métodos em uma aplicação *QuO*

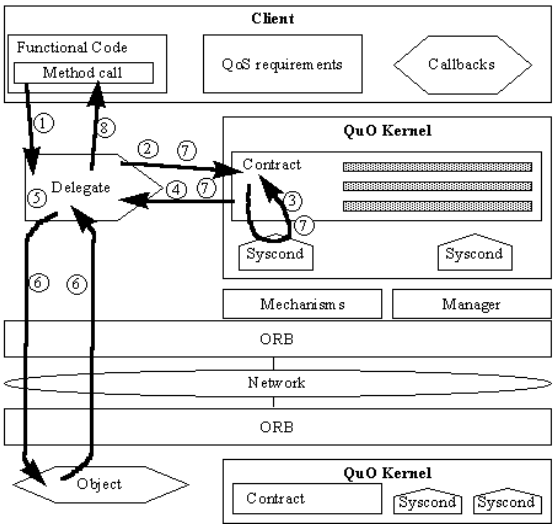
12

Visão Geral de *QuO*

- Objetos Delegados – provê uma interface idêntica à interface do objeto remoto, mas pode fazer verificação de contratos nas chamadas e retornos dos métodos
- Objetos de Contrato – especifica o nível de serviço esperado, as regiões de operação e acionamento de comportamento adaptativo
- Objetos de Condições de Sistema – faz a ligação entre os contratos e os recursos, mecanismos, objetos e *ORB´s*.

13

Visão Geral de *QuO*



1) Client makes a remote method call, which is delivered to delegate
2) Delegate evaluates contract
3) Contract gets the current values of system conditions to determine current region (if a region transition occurs, callback behavior is triggered)
4) Contract returns current regions
5) Delegate chooses behavior based on the current regions; default is to pass the method call through to the remote object
6) Remote object is invoked, performs its method, and returns a value
7) Delegate re-evaluates contract, contract gets current values of system conditions, and returns current regions
8) Delegate chooses behavior based on the current regions; default is to pass the return value back to the client

Detalhamento da chamada de método em *QuO*

14

Visão Geral de *QuO*

- Chamada é direcionada ao Objeto Delegado
- Delegado pode verificar o contrato
- O contrato possui um conjunto de regiões aninhadas que descrevem os possíveis estados de *QoS* no sistema
- As regiões são definidas de acordo com os valores de todas as propriedades de sistema avaliadas, através dos objetos de condições de sistema

15

Visão Geral de *QuO*

- Surge mais uma função a ser desempenhada na elaboração do sistema distribuído: a do desenvolvedor *QuO*
- O trabalho desse novo elemento é feito através de:
  - Uma coleção de *Quality Description Languages* (*QDL*) que descrevem contratos, objetos de condições de sistema e o comportamento adaptativo
  - Núcleo de *QuO* (*kernel*)
  - Geradores de código

16

QDL – Quality Description Languages

- *Contract Description Language* (*CDL*) – estabelece os padrões de uso esperado e os requisitos de QoS, define as regiões de condições de sistema e o comportamento adaptativo
- *Resource Description Language* (*RDL*) – descreve os recursos disponíveis no sistema e seus estados
- *Structure Description Language* (*SDL*) – define a estrutura interna de um objeto remoto

17

QDL – Quality Description Languages

- Elementos de um contrato:
  - Um conjunto de regiões aninhadas
  - Transições para cada nível de regiões e comportamento a ser disparado
  - Referência a objetos de condições de sistema, que podem ser parâmetros ou locais
  - *Callbacks* para notifiação de clientes e objetos
- Isso engloba os possíveis estados de um sistema, a informação que necessita ser monitorada e as ações a ser tomadas

18

QDL – Quality Description Languages

```
contract repl_contract(
    syscond  ValueSC ValueSCImpl ClientExpectedReplicas,
    callback AvailCB ClientCallback,
    syscond  ValueSC ValueSCImpl MeasuredNumberReplicas,
    syscond  ReplSC ReplSCImpl ReplMgr ) is

  negotiated regions are
    region Low_Cost : when ClientExpectedReplicas == 1 =>
      reality regions are
        region Low    : when MeasuredNumberReplicas < 1 =>
        region Normal : when MeasuredNumberReplicas == 1 =>
        region High   : when MeasuredNumberReplicas > 1 =>
        transitions are
          transition any->Low    :
   ClientCallback.availability_degraded();
          transition any->Normal :
   ClientCallback.availability_back_to_normal();
          transition any->High   :
   ClientCallback.resources_being_wasted();
        end transitions;
      end reality regions;
```

19

QDL – Quality Description Languages

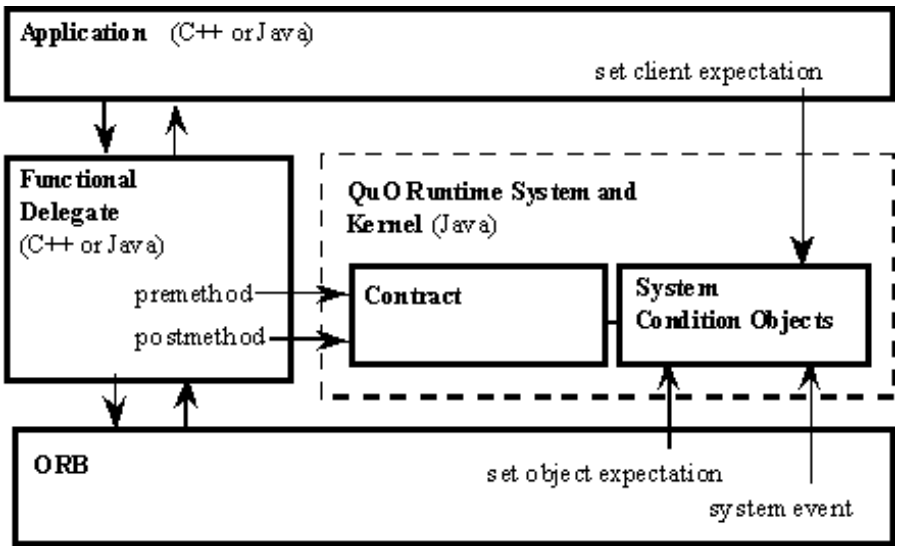
```
region Available : when ClientExpectedReplicas >= 2 =>
      reality regions are
        region Low    : when MeasuredNumberReplicas <
   ClientExpectedReplicas =>
        region Normal : when MeasuredNumberReplicas >=
   ClientExpectedReplicas =>
        transitions are
          transition any->Low    :
   ClientCallback.availability_degraded();
          transition any->Normal :
   ClientCallback.availability_back_to_normal();
        end transitions;
      end reality regions;
     transitions are
      transition Low_Cost->Available :

   ReplMgr.adjust_degree_of_replication(ClientExpectedReplicas);
      transition Available->Low_Cost :

   ReplMgr.adjust_degree_of_replication(ClientExpectedReplicas);
     end transitions;
  end negotiated regions;
end repl_contract;
```

20

Componentes Internos de *QuO*



Componentes Internos de *QuO*

- Ambiente de execução encapsulado em um objeto delegado que é executado na aplicação
- Contratos e condições de sistema executados em um núcleo (*kernel*) independente
- Comunicação entre o delegado e o núcleo feita através de *CORBA*
- Vantagens desse modelo
  - Contratos podem ser gerados na linguagem de implementação do núcleo
  - Comportamento assíncrono dos objetos de contratos e condições em múltiplos processos

22

Componentes Internos de *QuO*

- Núcleo: biblioteca de serviços, escritos em Java
- Serviço do Objeto *Factory*
  - Implementado usando reflexão em Java
  - Instancia, localiza e carrega objetos no núcleo
  - Núcleo mantém listagem de objetos usados por cada objeto delegado
- Serviço do Objeto *Contract Evaluator*
  - Agenda a avaliação de contratos
  - Avaliação por demanda de métodos
  - Avaliação disparada por observação

23

Componentes Internos de *QuO*

- Delegado: representante do objeto remoto, com comportamento adaptativo
- Código gerado a partir da *IDL* e *QDL*
- Descrição de Contratos
- Descrição Estrutural
  - Mecanismo de Seleção
  - Comportamentos alternativos
- Objeto implementado na linguagem do cliente, com suporte a C++ e Java

24

Componentes Internos de *QuO*

- *bind* substituído por *connect*
- *connect()*
  - Instancia um *CORBA proxy*
  - Localiza o núcleo do *QuO* ou cria um
  - Usa os serviços do objeto *Factory* para criar objetos
  - Inicializa e faz a ligação entre todos os objetos do ambiente de execução
- Objetos delegados podem estar aninhados
- Objetos do núcleo podem ser passados como parâmetros do *connect()*

25

Componentes Internos de *QuO*

- Objetos de Contrato
- Cada região do contrato é definida como uma sentença de predicado de todos os valores das condições
- O primeiro predicado verdadeiro determina a região atualmente ativa
- Pode-se usar um modificar *precedence* para manipular a ordem de avaliação
- Comportamentos de transição podem ser especificados

26

Componentes Internos de *QuO*

- Objetos de Sistema – uma única propriedade do sistema
  - Tempo passado desde a última chamada
  - Número de sessões ativas
- Objeto Java que implementa uma *interface* comum a todos os objetos de sistema
- Apenas *getValue()* na forma mais simples
- Objetos *CORBA* na forma mais complexa

27

Cenário de Desenvolvimento

- Programadores de Clientes
  - Usuários de objetos adaptativos
  - Trabalham de forma semelhante à de *CORBA*
  - Inicializam cliente com alguns comportamentos esperados
- Programadores de Objetos
  - Adicionam funcionalidade aos tipos de adaptação
- Programadores *QoS*
  - Contratos e condições de sistemas

28

Conclusões

- Aplicações distribuídas precisam adaptar-se a situações onde as condições de sistema são aquém das desejáveis
- Os desenvolvedores precisam de ferramentas para tratar esses casos de forma sistemática
- *QuO* propõe um modelo de arquitetura para prover *QoS* a aplicações distribuídas
- Os objetivos de *QuO* são complementares aos de *CORBA*, não de confronto

29