Suporte a Sistemas Abertos de Larga Escala para Empresas Virtuais

Joni S.Fraga, Ricardo J. Rabelo e Frank A. Siqueira

Departamento de Automação e Sistemas

Universidade Federal de Santa Catarina

Resumo

Neste artigo apresentamos uma plataforma para aplicações críticas construída sobre a arquitetura CORBA e seus suportes para tempo real (RT-CORBA), tolerância a faltas (FT-CORBA), e segurança (CORBASec). Resultados preliminares obtidos com a implementação da plataforma nos permitem levar em conta os requisitos temporais de um sistema multiagentes de escalonamento da produção em empresas virtuais. Este sistema é utilizado em aplicações de suporte ao comércio eletrônico inter-empresas, nas quais deve haver a troca de informação com o objetivo de executar de modo cooperativo um determinado processo de produção.

Abstract

This paper presents a platform for critical applications built upon the CORBA architecture and its supports for real-time (RT-CORBA), fault tolerance (FT-CORBA), and security (CORBASec). Preliminary results obtained with the implementation of this platform allow us to take into account the timing requirements of a multi-agent system for production scheduling in virtual enterprises. This system is used to build business-to-business applications, in which information must be exchanged in order to execute cooperatively a manufacturing process.

Palavras-Chave

CORBA, Tempo-Real, Tolerância a Faltas, Segurança, Sistemas de Produção, Sistema Multiagente, Empresas Virtuais.

1 Introdução

A união temporária de empresas de forma a colaborar na obtenção de um produto final, leva à formação das chamadas empresas virtuais [Browne95]. Empresas virtuais são formadas por núcleos originalmente independentes, mas que se unem em busca de um objetivo comum, e trocam informação de modo a coordenar as suas ações. Este cenário é ilustrado na Figura 1.

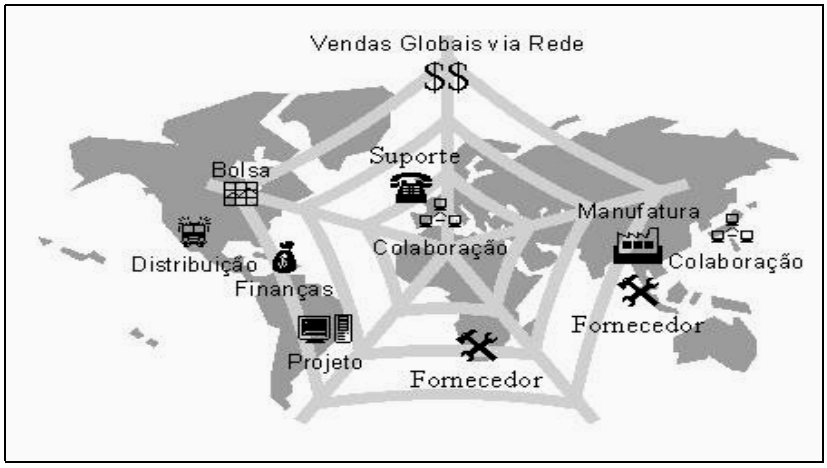

Figura 1 – O Processo de Produção em uma Empresa Virtual

O processo de produção em empresas virtuais engloba um significante leque de tecnologias, reunindo quantidades significativas de recursos *hardware* e *software* que se encontram distribuídos em sistemas computacionais de larga escala. O paradigma de empresas virtuais implica que as empresas devem empregar mecanismos de comunicação para trocar informações e para sincronizar eletronicamente as suas produções distribuídas, utilizando tecnologias como EDI (*Electronic Data Interchange*) e *Workflow*. Outros serviços de mais alto nível, tais como os de suporte ao projeto colaborativo de produtos e a monitoração em tempo real da entrega de encomendas, requerem infraestruturas extremamente robustas, seguras e eficientes para que sejam utilizados pelas empresas. No nível administrativo, uma série de ferramentas deve estar disponível para que a cooperação entre empresas possa ocorrer de maneira efetiva. Ferramentas de escalonamento de atividades são necessárias para que as ações das empresas envolvidas no processo de produção sejam efetuadas de forma coordenada, enquanto ferramentas de colaboração devem ser usadas para executar trabalho cooperativo.

Dos níveis citados acima, fica evidenciado que algumas das etapas do processo produtivo necessitam de serviços com garantias de desempenho, em particular a manufatura e o projeto colaborativo. Outras etapas precisam obter alta confiabilidade, principalmente no estabelecimento de transações entre as empresas e da empresa com o cliente. Adicionalmente, as transações financeiras e a informação trocada entre as empresas em relação ao projeto colaborativo e ao processo de manufatura devem utilizar-se de mecanismos de segurança para evitar interferências externas mal-intencionadas.

Devido à complexidade das tecnologias necessárias para operação de empresas virtuais e à heterogeneidade dos sistemas caracterizados como de larga escala, é necessário tornar disponível para as empresas um suporte computacional que facilite o emprego destas tecnologias. O suporte deve ainda permitir que interações entre empresas possam ocorrer de forma segura, confiável, e com um desempenho satisfatório. Adicionalmente, o suporte deve fornecer mecanismos para a especialização, no sentido de atender os requisitos de qualidade de serviço das empresas virtuais.

Ao longo deste artigo apresentaremos o SALE – um suporte a sistemas abertos de larga escala para empresas virtuais construído sobre a plataforma CORBA utilizando extensões para tempo-real, tolerância a faltas e segurança (Seção 2). Em seguida descreveremos o SC^2 – um sistema de escalonamento da produção em empresas virtuais desenvolvido sobre o SALE (Seção 3). Finalizando o artigo, analisaremos os resultados alcançados (Seção 4) e apresentaremos as nossas conclusões e perspectivas de evolução deste trabalho (Seção 5).

2 Suporte para Aplicações em Sistemas Abertos de Larga Escala: o Projeto SALE

O projeto SALE tem como objetivo desenvolver uma plataforma cuja arquitetura integra serviços com diferentes tipos de garantias. Esta arquitetura, adequada para novas aplicações que se configuram como sistemas globalmente distribuídos e que tratam com informações de diferentes naturezas, deve incluir certas funcionalidades e características que contribuam para a flexibilidade e a adaptação deste suporte às necessidades da aplicação. Para atender estes objetivos é necessário:

- Fazer uso extensivo de padrões abertos, que disponibilize um significante leque de tecnologias, permitindo soluções mais adaptadas à complexidade e à heterogeneidade de sistemas nestas aplicações normalmente definidas como de larga escala.
- Fornecer um ambiente conveniente e seguro para a especialização de serviços de suporte.

Esta arquitetura está centrada em ORBs (*Object Request Broker*) seguindo os padrões CORBA (*Commom Object Request Broker Architecture*). As especificações CORBA, proposta pela OMG (*Object Management Group*)[1] [OMG01], definem padrões para a comunicação entre objetos em um ambiente distribuído e heterogêneo – possibilitando entre outras coisas a interoperabilidade, a portabilidade do código das aplicações e o reuso de software. Este suporte deve possuir um caráter aberto adotando, além do CORBA, padrões como o POSIX e componentes de prateleira (*off the shelf*) de modo a propiciar a troca de informações e permitir que as empresas trabalhem de forma integrada.

[1] É uma organização formada por mais de 700 empresas com o objetivo de especificar um conjunto de padrões e conceitos para a programação orientada a objetos em ambientes distribuídos abertos.

2.1 Arquitetura do SALE

A plataforma SALE, ilustrada na Figura 2, tem como objetivo central integrar e conciliar mecanismos que forneçam diferentes garantias de desempenho, confiabilidade e segurança nas interações das aplicações. Estes atributos de qualidade devem ser especificados na forma de requisitos de qualidade de serviço (QoS). Os serviços de suporte são especializados a partir destes requisitos de QoS, envolvendo diferentes mecanismos na plataforma de suporte e em camadas subjacentes. Os mecanismos necessários em uma especialização são tornados disponíveis a partir de uma interface única, os *objetos de QoS* (figura 2).

Os requisitos de desempenho, confiabilidade e segurança da aplicação são fornecidos, fazendo uso das especificações *Real-Time CORBA* (RT-CORBA) [OMG98], *Faul-Tolerant CORBA* (FT-CORBA) [OMG99] e *CORBA Security* (CORBASec) [OMG98a], respectivamente. Portanto, serviços de objetos são propostos nesta plataforma tomando como base os *COSS* (*Commom Object Service Specification*) e serviços *ORB* definidos nestas especificações. Adicionalmente, dependendo dos requisitos da aplicação, podem ser usados suportes subjacentes de comunicação tempo-real, de comunicação de grupo e de comunicação segura como mostra a Figura 2.

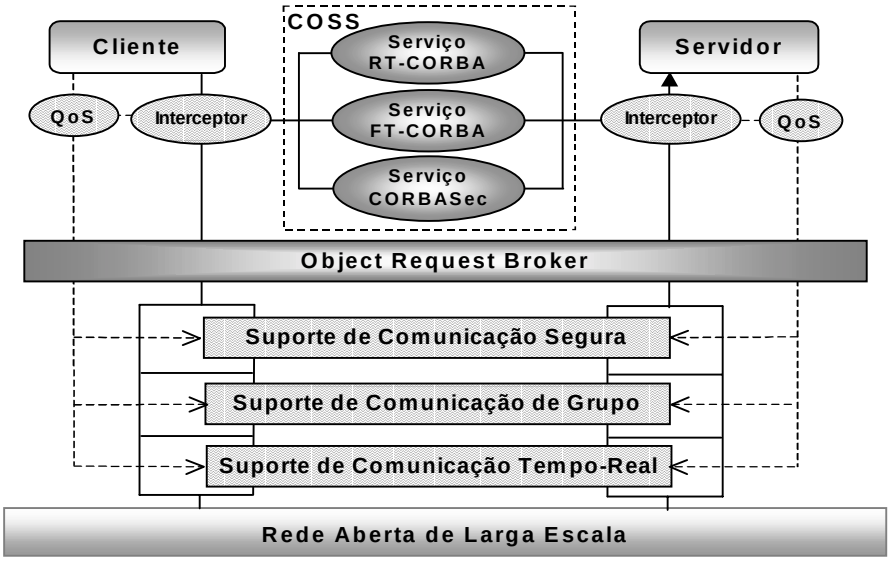

Figura 2 – A Arquitetura SALE

Suportes de *middleware* podem fornecer mecanismos que favorecem a especialização e a flexibilidade necessárias para a adequação à aplicação. O conceito de *interceptors*[2] é exemplo destes mecanismos nas especificações CORBA [OMG98b]. A interceptação em cada invocação de método através do ORB torna transparente a execução de políticas de controle sobre os métodos de aplicação. No SALE, as chamadas de objetos interceptadas são tratadas de forma diferenciada, usando entre os suportes disponíveis (ORBs RT-CORBA, FT-CORBA

[2] Interceptors são objetos logicamente interpostos na seqüência de invocação de um cliente a um servidor que tem a finalidade de desviar a chamada de maneira transparente, ativando assim um objeto de serviço associado, implementando um controle próprio para a aplicação em mãos.

e CORBASec), o mais adequado aos requisitos de QoS da aplicação. A seleção através destes requisitos influencia tanto a forma como o sistema local executa a aplicação quanto como a comunicação é concretizada.

Os ORBS e serviços de objeto CORBA que implementam os três requisitos – desempenho, confiabilidade e segurança – são instanciados separadamente, ou seja, a plataforma em uma especialização só tratará individualmente um destes requisitos. Ao longo da evolução do projeto tentaremos compatibilizar requisitos que não se demonstrem incompatíveis conceitualmente. Este processo deve seguir o caminho adotado pela OMG, que tem mostrado preocupação em particular na obtenção de compatibilidade entre as tecnologias FT-CORBA e CORBASec [OMG99]. Neste texto, nos itens subseqüentes, apresentamos as nossas experiências iniciais no projeto SALE.

2.2 Mecanismos para a Especificação e a Obtenção de QoS

Ao se considerar aplicações que apresentam uma forma tão diversificada de dados, fica evidenciado a necessidade de mecanismos flexíveis de especificação dos requisitos de QoS que expressem as diferentes qualidades usando conceitos inteligíveis ao nível de aplicação [Siqueira99]. A plataforma SALE fornece mecanismos tanto estáticos quanto dinâmicos para especificação de QoS. A linguagem QSL (*Quality Specification/Scripting Language*), que se encontra em desenvolvimento, permite que os requisitos de QoS de uma aplicação sejam expressos estaticamente. *Scripts* de mapeamento de parâmetros especificam como os diferentes formatos de parâmetros suportados são mapeados nos recursos do suporte. Os diferentes requisitos são definidos a partir do servidor tomando como base a execução de seus serviços e implementados a partir do cliente sobre chamadas remotas fim-a-fim.

Requisitos podem ser associados a chamadas individuais, a todas as chamadas do mesmo método ou a todas as chamadas a um objeto. Os *Objetos QoS* são gerados tendo como base uma especificação em QSL própria para a aplicação. Conforme mostrado na Figura 2, objetos QoS são localizados a nível de aplicação, juntamente com o objeto cliente e com o objeto servidor, permitindo que o suporte computacional seja configurado em tempo de ligação (*binding*) de modo a atender os requisitos da aplicação. Esta abordagem é compatível com os modelos RT-CORBA, FT-CORBA e CORBA*Sec*, onde os objetos de serviço e mecanismos que atuam em tempo de execução nas interações cliente/servidor são criados ao se estabelecer *bindings* entre objetos. Os Objetos QoS são consultados por interceptadores durante a invocação de modo a obter os valores dos parâmetros de QoS, para que requisitos de QoS sejam observados durante a execução da aplicação distribuída. Objetos QoS permitem ainda que os valores atribuídos aos parâmetros de QoS (por exemplo, a prioridade CORBA de um método) sejam modificados em tempo de execução através da interface mostrada na Figura 3.

O cumprimento dos requisitos de QoS da aplicação depende da disponibilidade de recursos dos sistemas operacionais e dos suportes de comunicação envolvidos. Os parâmetros

de QoS da aplicação são mapeados nestes recursos subjacentes e reservados, garantindo à aplicação os níveis desejados de qualidade.

```
struct Property{          // Do Serviço de Propriedades do CORBA
  string property_name;
  any    property_value;
};
typedef sequence<Property> Properties;

module SALE {             // IDL do SALE
  interface Qos {
    exception ParamEx { Properties qos_params; };
    exception ParseEx { short num_linha; string qsl_expr; };
    attribute Properties myQos;
    void readQosSpec (in string filename)       raises (ParamEx,ParseEx);
    void setParam (in string name, in any val)  raises (ParamEx);
    void getParam (in string name, out any val) raises (ParamEx);
    void configQos()                            raises (paramEx);
  };
};
```

Figura 3 – IDL do Objeto QoS do SALE

2.3. FT-CORBA

Os objetos e mecanismos definidos a partir das especificações FT-CORBA [OMG99] buscam garantir os requisitos de confiabilidade da aplicação relacionados à sua execução. Nestas especificações, introduzidas recentemente, são apresentados os objetos de serviço, que fornecem no *middleware*, funcionalidades básicas para a construção e gerenciamento de servidores tolerantes a faltas. Foi introduzido um conjunto de interfaces de serviços envolvendo gerenciamento de replicação, detecção de falhas e a retomada de serviços no sistema (serviços de *gerenciamento de replicação*, *gerenciamento de faltas* e *gerenciamento de recuperação e logging*). Estas especificações FT-CORBA definem também as técnicas de replicação que podem ser suportadas pelo *middleware* (*Replicação ativa*, *Replicação passiva fria* e *Replicação passiva quente* [OMG99]).

A OMG — na sua característica de atender diferentes abordagens de tolerância a falhas – se limita em definir interfaces de serviço genéricas, omitindo-se por exemplo da padronização de serviços de comunicação de grupo confiável, base fundamental para a implementação de técnicas de replicação ativa. É enfatizado nas especificações, para este caso, o uso de soluções proprietárias. Todavia, para garantir um mínimo grau de interoperabilidade, os sistemas de comunicação de grupo devem adotar a IOGR (*Interoperable Object Group Reference*), a referência de grupo de objetos definida em [OMG99]. A IOGR é uma extensão da IOR (*Interoperable Object Reference*), de um simples objeto, que permite a um cliente referenciar um grupo de objetos como a uma entidade única.

Os nossos esforços em relação ao FT-CORBA correspondem à implementação das especificações no projeto *GroupPac* [Lau00] que ainda não foram integradas ao SALE. O *GroupPac* pode ser obtido no endereço http://www.lcmi.ufsc.br/grouppac.

2.4. CORBASec

O modelo de segurança introduzido no *CORBASec* estabelece vários procedimentos envolvendo a autenticação e a verificação da autorização na invocação de um método remoto [OMG98a]. O modelo CORBA de segurança relaciona objetos e componentes de quatro níveis de um sistema: os objetos cliente e o servidor em nível de aplicação; em nível de *middleware* os objetos de serviço (*COSS*), serviços ORB (interceptadores) e o núcleo do ORB; componentes de tecnologia de segurança em nível de serviços de segurança subjacentes e, por fim, componentes de proteção básica (hardware e sistemas operacionais locais).

No CORBASec, para ativação dos Objetos de serviço são definidos dois tipos de interceptadores que atuam durante uma invocação de método: o interceptador de controle de acesso (*Access Control Interceptor*) e o interceptador de chamada segura (*Secure Invocation Interceptor*). O primeiro interceptador que atua em nível mais alto, provoca um desvio para objetos de serviço que implementam as políticas de controle de acesso. O interceptador de chamada segura que concretiza uma interceptação de mais baixo nível, desvia para objetos de serviço que encapsulam os serviços subjacentes da camada de tecnologia, de modo a fornecer propriedades de integridade e de confidencialidade nas transferências de mensagens correspondentes à invocação.

A nossa experiência em relação às especificações *CORBASec* corresponde às implementações no projeto *JaCoWeb* [Westphall99] que também ainda não foi integrada ao SALE no estágio atual. Informações do *JaCoWeb* podem ser obtidas na WEB, a partir de http://www.lcmi.ufsc.br e http://www.lrg.ufsc.br/JaCoWeb/.

2.5. RT-CORBA

As especificações RT CORBA 1.0 [OMG98] descrevem *threads* como as entidades escalonáveis do sistema. Interfaces são especificadas para gerir *threads*, o que propicia uma grande portabilidade às aplicações. *Threads* podem ser definidas e controladas de forma uniforme, independentemente do suporte de *threads* do sistema operacional e de seus mecanismos de escalonamento. As políticas de escalonamento são implementadas através da atribuição de prioridades das *threads*. *Pools* de *threads* e filas de requisições são recursos que aplicações podem manipular diretamente. Um *pool* de *threads* pode ser criado para processar requisições de métodos recebidas por um servidor. Uma fila de requisição pode ser criada e associada com um *pool de threads* .

A *prioridade CORBA* é um tipo de prioridade global que pode atravessar diferentes ORBs, dentro de mensagens de invocação. Dependendo da abordagem de escalonamento, essa prioridade pode ser usada pelos servidores para ordenar as requisições recebidas. O documento RFP introduz dois modos de utilização: o *client-priority propagation* e o *server-set priority*. Em ambas as estratégias, a prioridade CORBA da invocação deve ser mapeada em prioridades de *threads* no sistema operacional antes da execução do método.

Durante um *binding* explícito, vários passos devem ser tomados para interconectar objetos, tais como localizar objetos destino e iniciar estruturas de dados usadas na comunicação entre objetos. No lado do cliente, o *binding* pode ser usado, por exemplo, para a seleção de um protocolo de transporte apropriado e de um valor de *timeout* para o bloqueio do cliente. No lado do servidor, o *binding* permite a alocação dos recursos necessários na execução das requisições, tais como *threads* e filas. Estas especificações foram definidas considerando escalonamentos dirigidos a prioridades fixas. Com o objetivo de adicionar suporte para escalonamento dinâmico ao RT-CORBA, a OMG lançou um novo RFP que em breve deve resultar na evolução do RT-CORBA para a versão 2.0.

RT-CORBA no Projeto SALE

O RT-CORBA define objetos *COSS* e extensões a um ORB convencional no sentido de ordenar filas de requisições segundo políticas de prioridades fixas. As políticas de tratamento de prioridade que interessam no projeto SALE são as do tipo *client-propagated*. Ou seja, no SALE os requisitos de desempenho do cliente são implementados usando prioridades CORBA, que podem refletir *deadlines* ou outras restrições temporais.

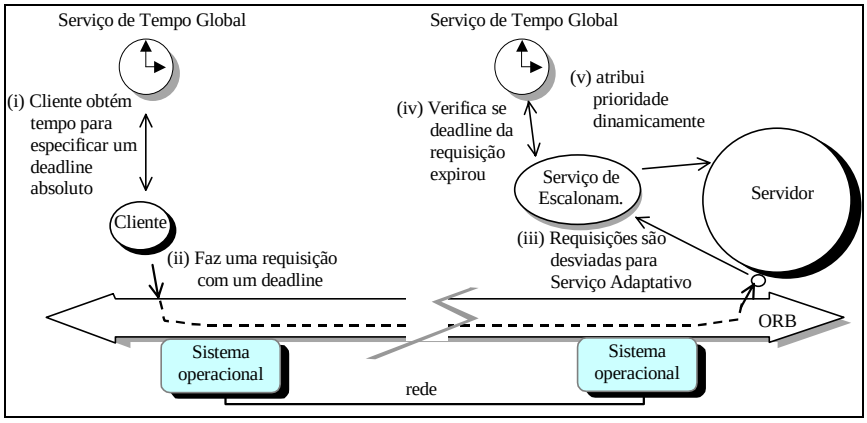

Figura 4 - A Invocação de Tempo-Real

A Figura 4 apresenta um cenário de uso do modelo. A abordagem de escalonamento é implementada por um objeto de serviço CORBA presente em todos os nós servidores – o Serviço de Escalonamento. Interceptadores no servidor são usados para desviar as requisições para o Serviço de Escalonamento de forma transparente. Esse serviço é responsável por implementar as políticas de escalonamento que atuam sobre as invocações a métodos no servidor. A pré-alocação de um *pool de threads* no servidor para executar requisições de clientes aumenta o desempenho do mesmo. O suporte de comunicação de tempo real depende dos serviços subjacentes da tecnologia de rede disponível.

Ocorrências de sobrecarga são tratadas usando técnicas adaptativas somente no nó do servidor. O cliente atribui um deadline absoluto ou uma prioridade global para a invocação do método antes de enviar a requisição, caracterizando então uma *invocação de tempo real*. Quando usados valores de *deadlines* absolutos, é necessário uma base de tempo global, que permita uma visão coerente em todo o sistema sobre estes valores de tempo. Em [Fraga98],

um *Serviço de Tempo Global* estende as especificações do Serviço de Tempo CORBA, fornecendo uma base de tempo global. O Serviço de Tempo Global pode ser implementado em um sistema largo e disperso, usando receptores GPS.

No primeiro protótipo implementado, o modelo de invocação de tempo real considera que cada invocação é executada completamente, mesmo após o *deadline*. Não existem descartes ou abortos de execuções de requisição, apesar de conceitualmente um método com deadline não apresentar normalmente benefício máximo quando executado após o *deadline*. Portanto, o passo (iv) na Figura 4 não é implementado. Neste protótipo, conforme será discutido posteriormente (item 3.2), o valor de deadline (*deadline* absoluto) é definido no relógio do servidor a partir de um intervalo de tempo (*deadline* relativo) propagado pelo cliente. Esta solução dispensa a necessidade de um serviço de tempo global e de receptores GPS.

Nas nossas implementações do suporte RT-CORBA do projeto SALE utilizamos o TAO – um ORB de alto desempenho para tempo real desenvolvido na Universidade de Washington em Saint Louis [Schmidt97], que tem como alvo ambientes deterministas. O ORB TAO teve grande influência no desenvolvimento do padrão RT-CORBA aprovado pela OMG. O TAO possui algumas facilidades próprias para programação tempo real, e seu código-fonte é aberto e disponível na Internet. Isso nos possibilitou implementar alguns mecanismos necessários na tentativa de conformidade com as especificações RT-CORBA (por exemplo, foram implementados mecanismos para desvios de requisições, porque a versão do TAO utilizada na implementação não possui *interceptors*).

3 O SALE como Suporte de Aplicações para Empresas Virtuais

Como já comentado, no cenário atual onde as empresas buscam freneticamente aumentar seus níveis de competitividade, o comércio eletrônico tem surgido como uma das mais importantes apostas. Embora a parte mais "popular" seja veiculada constantemente como a "B2C" (*Business-to-Consumer*) na qual o cliente final efetua a sua transação diretamente com a empresa prestadora do serviço, ela é praticamente insignificante quando comparada à forma "B2B" (*Business-to-Business*). Este é o cenário no qual as empresas realizam todo tipo de transação entre si (compra/venda, monitoramento de execução, planejamento cooperativo de compras, desenvolvimento colaborativo de produto, etc.) e onde as questões de robustez, segurança e desempenho são cruciais. É importante notar que no cenário B2B o número de transações "simultâneas" costuma ser bem pequeno se comparado ao cenário B2C.

No contexto do B2B, várias formas de organização podem existir, genericamente "encapsuladas" com o termo Organizações Virtuais (OV – *Virtual Organizations*). Sucintamente, reflete um conjunto de empresas que estão disponíveis (i.e. tem infraestrutura de comunicações e serviços) para trabalhar em rede visando atender oportunidades de negócio. Dependendo do tipo de empresas e negócio, diferentes formas de OV são usadas dentro desse conceito, nomeadamente as de Empresas Virtuais (EV - *Virtual Enterprises*),

Empresas Estendidas (*Extended Enterprises*) e Cadeias de Fornecimento (*Supply Chain*). Nestas, há uma noção implícita da necessidade de partilha de recursos e informações – e de forma coordenada – para que o conjunto de transações envolvido no negócio seja não "apenas" realizado, mas com a *qualidade* acordada. Nesse contexto, conforme já citado anteriormente a *qualidade* tem diversas facetas, dependendo do nível considerado.

A plataforma SALE vem a atender esses requisitos, focando principalmente no cenário de EVs – o mais complexo e abrangente – oferecendo um conjunto de serviços de infraestrutura para permitir uma adequada execução de serviços no nível aplicação.

3.1 O Arcabouço da Aplicação

A aplicação sendo desenvolvida – chamada *Sistema SC²* – é constituída a partir de um sistema multiagente[3] de suporte à decisão que visa assistir o gestor da empresa virtual na monitoração da execução de um "processo de negócios" [Rabelo01]. O sistema *SC²* engloba resultados de três recentes projetos internacionais onde a UFSC esteve envolvida: PRODNET-II (http://www.uninova.pt/~prodnet), MASSYVE (http://www.gsigma-grucon.ufsc.br/english/projects.htm#Massyve) e DAMASCOS (http://bart.inescn.pt/~damascos).

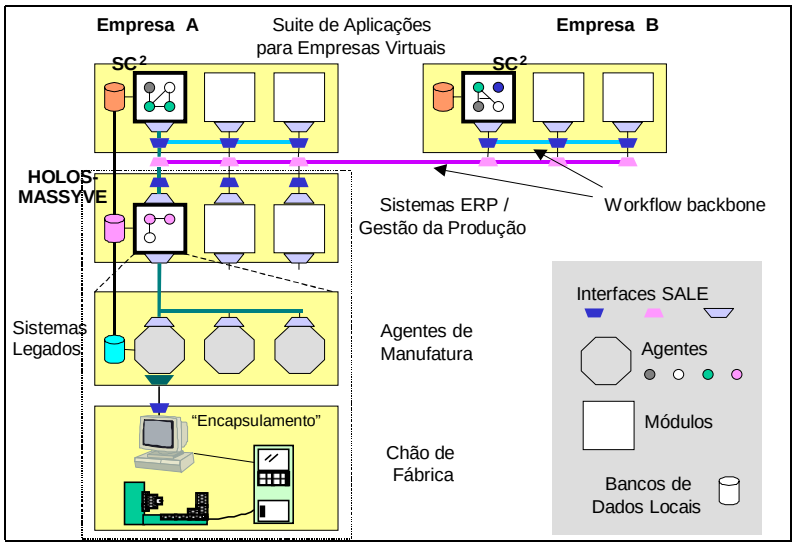

Figura 5 – Arcabouço da plataforma SALE+SC²

A plataforma SALE é vista aqui como base genérica de serviços de infraestrutura na qual serviços de aplicação, no caso sendo estudado os do SC², podem ser suportados no que se refere a comunicação. A Figura 5 ilustra o arcabouço SALE+SC². No caso ilustrado, é representada uma empresa virtual hipoteticamente composta por duas empresas ("A" e "B"). A coordenação desta empresa virtual se dá através de instâncias do *SC²* nas duas empresas componentes. A representação virtual de uma empresa componente ("A" ou "B") é composta por uma *suite* de módulos com serviços de alto nível. O SC² é um destes módulos.

[3] Sucintamente, no contexto em questão, um sistema multiagente é uma agregação de processos computacionais ("agentes") que, com base em variados graus de autonomia, interagem entre si visando atingir a um objetivo comum.

Toda a comunicação intra e inter-suite é feita através de um sistema de *workflow*, com as informações trocadas em formato XML no *backbone* disponibilizado. Por exemplo, sempre que um agente do SC^2 de uma empresa quer alguma informação, ele previamente se inscreve numa lista (por "assunto") do *workflow backbone*; e sempre que um agente tem a informação desejada, ele publica a mesma no *backbone* e o mecanismo de *workflow* a envia para os agentes clientes da lista. A Figura 6 mostra a IDL do mecanismo de *workflow* suprido pelo SALE, que é construído sobre o serviço de notificação do CORBA [OMG00].

```
#include "UserIdentifier.idl"
module WorkFlowBackbone {
typedef sequence <string> string_array;
module Administration {
  module UserAdministration {
    interface PrivateUser  {
        attribute string password;
        void destroy();
    };
    interface User;
    interface PublicUser : UserIdentifier, ChannelAdministration::PublicProxy {
        m_ChannelAdmin_def getChannelAdmins ();
        void setPassword (in string password);
        string_array listParentGroups();
        void setPushGroup (in UserGroup grp);
        void setPushUser (in User usr);
        void pushGroup (in UserGroup grp, in string msg);
        void pushUser (in User usr, in string msg);
    };
    interface User : PrivateUser, PublicUser {};
  };
};
```

Figura 6 – IDL do *Workflow Backbone* usada pelos Agentes SC^2

Note que plataformas de suporte à interoperação em uma empresa virtual deverão encapsular as plataformas legadas com as quais as empresas componentes já trabalham. Portanto, e em termos muito gerais, poder-se-ia afirmar que a plataforma *SALE + SC^2* funciona como uma espécie de "plug-in" para uma empresa realizar digitalmente transações com outras empresas (igualmente "plugadas"). Os sistemas legados sistemas (ERP, etc.) são encapsulados na forma de objetos CORBA (agentes, no caso) através do SALE. Portanto, não há como deixar de se fazer alguma "re-engenharia" nos sistemas legados [C.-Matos99]. Por exemplo, se a empresa componente B desejar monitorar o andamento do seu pedido junto ao fornecedor empresa componente A, vários sistemas legados estarão envolvidos (tais como acessar a informação do chão de fábrica, atualizar no banco de dados, encapsular o conteúdo desejado no formato adequado, criptografar, e enviar a informação). Estes serviços, quando disponibilizados a nível de objetos CORBA, representam a re-engenharia citada.

O sistema HOLOS-MASSYVE [Rabelo00], também multiagente, atua na coordenação da produção de um certo chão de fábrica de uma empresa componente (sistema de *scheduling* em uma célula de produção). Do ponto de vista de cenário B2B, esse sistema atua como interlocutor entre as necessidades da empresa virtual com as disponibilidades do chão de fábrica de uma empresa componente. O sistema HOLOS-MASSYVE foi concebido para reagir em função de indisponibilidades não previstas dos equipamentos industriais e de pedidos aleatórios dos clientes. Para tal, suporta uma negociação entre seus agentes ("agentes

de manufatura" da Figura 5) visando criar, dinâmica e temporariamente, a mais adequada composição de equipamentos para um dado plano de produção.

Dada a necessidade de se supervisionar permanentemente o estado do chão de fábrica, o sistema HOLOS-MASSYVE requer uma comunicação direta – sob restrições de tempo real – com os equipamentos de forma a poder reagir à problemas e, assim, manter coerente o plano de produção. Todas essas comunicações inter-agentes no SALE se dão na forma tradicional cliente/servidor, segundo o modelo RT-CORBA indicado na Figura 4.

As interfaces dos agentes de manufatura dependem do tipo de atividade executada pelos sistemas legados encapsulados por estes agentes. A Figura 7 mostra um exemplo de interface IDL do agente de um controlador.

```
interface Controller {  // Agente de manufatura do HOLOS/MASSYVE
    void load_program (in short machine, in string filename);
    void start_program (in short machine);
    void stop_program (in short machine);
    void get_status (in short machine, out short status);
};
```

Figura 7 – Exemplo de IDL de um Agente HOLOS/MASSYVE

3.2 Requisitos de Sistema

Com base na descrição exposta na seção anterior, cada nível da hierarquia dentro de uma empresa possui diferentes requisitos de qualidade de serviço. Dessa forma, por exemplo, os agentes SC^2 que representam empresas componentes, dialogam com seus pares assumindo requisitos de tempo-real sobre o mecanismo de *workflow*. Já nos agentes de mais baixa hierarquia, aqueles que são ligados aos controladores das máquinas do chão de fábrica, as restrições de tempo real são expressas no modelo cliente-servidor conforme enfatizado nas especificações RT-CORBA.

QoS sobre o Mecanismo de Workflow usado pelos Agentes SC^2

Para os agentes SC^2, ou seja, no nível administrativo da EV, o envio de eventos através do *workflow backbone* é efetuado pelo serviço de notificação. A entrega de eventos segue a política de melhor esforço (*best-effort*), pois a este nível os requisitos temporais são menos restritivos. Os requisitos de QoS previstos no serviço de notificação do SALE e usados pelo *workflow backbone* são descritos tomando como base os seguintes parâmetros:

- *Reliability*: forma de entrega do evento (*best-effort* ou *persistent*);
- *Priority*: prioridade de entrega do evento;
- *Timeout*: o tempo de validade do evento, a partir do qual ele pode ser descartado.

Os valores dos parâmetros de QoS são obtidos estaticamente a partir da especificação em QSL, mostrada na Figura 8, ou em tempo de execução caso os valores sejam alterados através do objeto QoS, usando a interface descrita na Figura 3.

No envio de eventos são observados *timeouts* com ordem de grandeza de algumas horas ou dias, devido à natureza das atividades executadas no nível SC^2. O valor dos parâmetros é

inserido pelo SALE no corpo do evento ao interceptar a chamada (ver Figura 2). Em seguida, o evento é transmitido pelo serviço de notificação, levando em conta o *timeout* estipulado.

```
#include "Notification.QSL"; // Define parâmetros de QoS do Serviço de Notificação
QoS User: Notification {     // Objeto User emprega o serviço de notificação
  reliability = BEST-EFFORT; // Eventos serão entregues com melhor esforço
  priority = 5000;           // Prioridade de entrega dos eventos
  pushGroup::timeout = 7200; // Timeout no envio de eventos ao grupo (2 horas)
  pushUser ::timeout = 3600; // Timeout no envio de eventos ao usuário (1 hora)
};
```

Figura 8 – QSL do *Workflow Backbone*

QoS dos Agentes de Manufatura

No caso dos agentes HOLOS/MASSYVE, os requisitos temporais são observados pela plataforma na execução das chamadas, utilizando o suporte do RT-CORBA. Os requisitos de QoS dos agentes de manufatura são descritos com base nos seguintes parâmetros:

- *Timeout*: o tempo máximo de espera do cliente pelo retorno da chamada;
- *Deadline*: o tempo máximo de duração da execução do método no servidor;
- *Priority*: prioridade de execução do método no servidor.

Os valores dos parâmetros são obtidos a partir da especificação em QSL (Figura 9), e podem ser alterados em tempo de execução através do objeto QoS do SALE (Figura 3).

```
#include "RT-CORBA.QSL";          // Define parâmetros de QoS do RT-CORBA
QoS Controller: RT-CORBA {        // Objeto Controller usa o suporte RT-CORBA
  start_program::timeout = 0.005; // Timeout para iniciar a operação (5ms)
  stop_program::timeout = 0.005;  // Timeout para parar a operação (5ms)
  deadline = T2D(timeout);        // Deadline calculado em função do timeout
  priority = D2P(deadline);       // Prioridade calculada em função do deadline
};
```

Figura 9 – QSL de um Agente HOLOS/MASSYVE

As chamadas aos agentes de manufatura são efetuadas utilizando o modelo de programação descrito previamente na Figura 4, usando os valores de *timeout* e *deadline* correspondentes. O *deadline* é calculado em função do *timeout* estipulado, considerando o tempo gasto no processamento no cliente e na transmissão dos dados de entrada e saída. No momento da chamada, o valor calculado do *deadline* é obtido pelo interceptador a partir do objeto QoS no lado do cliente, inserido no contexto da chamada, e enviado na requisição. O *deadline* corresponde a um intervalo de tempo que será transformado em valor de tempo usando o relógio do servidor, o que nos libera da necessidade de sincronização de relógios ou de servidores GPS. A requisição é interceptada no lado do servidor e desviada para o serviço de escalonamento de tempo real (Figura 4). Um valor de prioridade é gerado em função do *deadline* da chamada e da política de escalonamento tempo real (no caso, *EDF* [Liu73]). Este valor é inserido na requisição no campo correspondente à prioridade CORBA, e usado no POA (*Portable Object Adapter*) para definir o valor da prioridade da *thread* de despacho do método no servidor.

4 Resultados Obtidos e Trabalhos Relacionados

Baseado nos resultados até então obtidos com o uso da plataforma SALE no suporte à coordenação de empresas virtuais e na coordenação de células de manufatura, o que se pode preliminarmente afirmar é a adequação do SALE nas suas extensões do RT-CORBA ao cenário de aplicação descrito. O modelo de programação apresentado na Figura 4 – que não define um protocolo determinista de comunicação e usa políticas *Best-Effort* de escalonamento tempo real – é usado em vários níveis da aplicação descrita, nas comunicações Cliente/Servidor. Embora as características não favoreçam a garantia de requisitos de tempo real, o uso do mesmo em níveis mais baixos não representa em problema para a aplicação em questão. Em manufatura discreta, onde as operações (usinagem, montagem, empacotamento, etc.) têm datas de início e fim planejadas dentro de um certo tempo *padrão* de execução, desvios da ordem de minutos são comuns de ocorrerem em um turno de trabalho. Ou seja, soluções não deterministas são aceitáveis nesse contexto de aplicação. Na manufatura contínua, devido à sua maior complexidade, estas soluções não são aceitáveis. Como os processos envolvidos usualmente lidam com tarefas críticas, tais como ajustes de pressões, abertura / fechamento de válvulas, leituras precisas de volumes, etc., o perfeito controle dos tempos de execução é quase impossível de ser feito com soluções *Best-Effort*. Neste cenário, décimos de segundo de tempo de execução atrasado ou adiantado podem incorrer em grandes danos no processo produtivo.

A evolução dos suportes para especificação e obtenção de serviços com garantias de QoS tem se pautado principalmente no fornecimento de mecanismos para serviços de comunicação com restrições temporais. No entanto, alguns suportes tem apresentado requisitos de QoS relacionados com a confiabilidade e a segurança. Ao nível de especificação de QoS, destacamos o trabalho dos HP Labs [Frolund98] envolvendo requisitos tanto para desempenho quanto para confiabilidade e segurança. Este trabalho baseia-se em construções de linguagem que são utilizadas para definir requisitos de qualidade a serem observados nos serviços estaticamente. Este trabalho ainda não define mecanismos para obtenção dos requisitos de QoS. Uma série de outros trabalhos em andamento busca permitir que aplicações construídas sobre o CORBA sejam dotadas de interfaces para especificação de requisitos de QoS, entre os quais destacamos o projeto MMN [Waddington96] e o *QuO* (*Quality of Service for CORBA Objects*) [Zinky97]. A própria OMG estuda a possibilidade de iniciar o processo de padronização das interfaces do ORB para especificação de QoS.

Os aspectos de tolerância a faltas e segurança propostos na arquitetura SALE ainda não são considerados ainda nas implementações atuais. Com a evolução do SALE e do SC^2, estes aspectos devem ser contemplados.

5 Conclusões

O projeto SALE se utiliza das especificações CORBA – nos seus modelos para tolerância a falhas, tempo real e segurança – na definição de mecanismos próprios para aplicações de Empresas Virtuais. Esta plataforma é especializada segundo requisitos de QoS, adaptando-se à natureza dos dados da aplicação tratada. No texto apresentado são mostrados os nossos primeiros resultados envolvendo questões de tempo real. Dois mecanismos de comunicação (*workflow* e Cliente/Servidor) definidos a partir das especificações RT-CORBA foram utilizados em diferentes níveis da hierarquia da aplicação. Estes mecanismos são veículos para a implementação de diferentes requisitos de qualidade de serviço necessários na complexa aplicação apresentada.

Bibliografia

[Browne95] J. Browne, P. Sackett, J. Wortmann, **"Future Manufacturing Systems: Towards the Extended Enterprise"**, Computer in Industry, Special Issue on CIM in the Extended Enterprise, Vol. 25(3), pp. 235-254, 1995.

[C.-Matos99] L. M. Camarinha-Matos, H. Afsarmanesh, **"Further Developments in Virtual Enterprises"**, em "Infrastructures for Virtual Enterprises: Networking Industrial Enterprises", L. M. Camarinha-Matos, H. Afsarmanesh (Eds.), Kluwer Academic Publishers, Outubro de 1999.

[Fraga98] J.S.Fraga, J.-M. Farines, C. Montez; "**Um Serviço de Tempo Global para Sistemas Distribuídos de Larga Escala**", 16° SBRC – Simpósio Brasileiro de Redes de Computadores, Rio de Janeiro-RJ, Maio de 1998.

[Frolund98] S. Frolund, J. Koistinen, **"Quality of Service Aware Distributed Object Systems"**, White Paper, Hewlett-Packard, 1998.

[Liu73] C.L. Liu, J.W.Layland, "**Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment**", Journal of the ACM, Vol. 20(1), January 1973.

[Lung00] L. C. Lung, J. S. Fraga, J.-M. Farines, R. S. Oliveira, **"Experiências com Comunicação de Grupo nas Especificações Fault Tolerant CORBA"**, Anais do XVIII Simpósio Brasileiro de Redes de Computadores, Belo Horizonte – MG, Maio de 2000.

[OMG98] Object Managemet Group, **"Realtime CORBA 1.0: Revised Submission"**, OMG Document orbos/98-12-10, Dezembro de 1998.

[OMG98a] OMG, **"Security Service: v1.2 Final"**, Object Management Group, Document 98-01-02, in CORBAServices Nov. 1998.

[OMG98c] Object Management Group, "**Portable Interceptor**", OMG Document Orbos/ 98-05-05, Maio de 1998.

[OMG99] Object Management Group, **"Fault-Tolerant CORBA"**, Joint Revised Submission Document orbos/99-12-08, December 1999.

[OMG00] Object Management Group "**Notification Service Specification**", OMG Document formal/00-06-20, June 2000.

[OMG01] Object Management Group "**The Common Object Request Broker: Architecture and Specification**", Revision 2.5, OMG Document formal/01-09-01, September 2001.

[Osório99] A. L. Osório, M. M. Barata, P. Gibon, **"Communication Infrastructure Requirements in a VE"**, em "Infrastructures dor Virtual Enterprises: Networking Industrial Entrerprises", L. M. Camarinha-Matos e Hamideh Afsarmanesh (Eds.), Kluwer Academic Publishers, Outubro de 1999.

[Osório99a] A. L. Osório, C. Antunes, M. M. Barata, **"The PRODNET Communication Infrastructure"**, em "Infrastructures for Virtual Enterprises", L. M. Camarinha-Matos, H. Afsarmanesh (Eds.), Kluwer, Outubro de 1999.

[Rabelo00] R. J. Rabelo **"Interoperating Standards in Multiagent Manufacturing Scheduling Systems"**, International Journal of Computer Applications in Technology (IJCAT), 2000.

[Rabelo01] R. J. Rabelo, R. V. Vallejos, **"A Semi-Automated Brokerage for a Virtual Organization of Mould and Die Industries in Brazil**", First IFPI Conference on E-Commerce, E-Business, E-Government, Zurique, Suiça, Outubro 2001.

[Schmidt97] D. C. Schmidt et al, **"TAO: A High Performance Endsystem Architecture for Real-Time CORBA"**, IEEE Communications Magazine, 14(2), Feb. 1997.

[S.-Costa01] S. Simões-Costa, R. J. Rabelo "**Supporting the Creation of Virtual Enterprises using Mobile Agents"**, submetido ao PRO-VE'2002 – 3rd IFIP Working Conference on Infrastructures For Virtual Enterprises, 2002.

[Siqueira99] F. Siqueira, V. Cahill **"Delivering QoS in Open Distributed Systems",** Proceedings of the 7th IEEE Workshop on Future Trends in Distributed Computing Systems, Cidade do Cabo, África do Sul, Dez. 1999.

[Waddington96] D. Waddington, G. Coulson and D. Hutchinson, **"Specifying QoS for Multimedia Communications within a Distributed Programming Environment"**, Proceedings of the 3rd International COST237 Workshop: Multimedia Telecommunications and Applications, Lecture Notes in Computer Science, Vol. 1185, pp104-130, Barcelona, Espanha, Nov. 1996.

[Westphall99] C. M. Westphall, J. S. Fraga, **"Authorization Schemes for Large-Scale Systems based on Java, CORBA and Web Security Models"**, Proceedings of the IEEE International Conference on Networks – ICON'99, Brisbane, Australia, Sep. 1999.

[Zinky97] J. Zinky, D. Bakken, R. Schantz **"Architectural Support for Quality of Service for CORBA Objects"**, Theory and Practice of Object Systems, Vol. 3(1), Janeiro de 1997.