Um *framework* de simulação voltado para interoperabilidade

Denis Gonçalves Cople[1]
Eduardo Siqueira Brick[2]

Resumo: Os sistemas de informática vêm evoluindo muito e várias soluções prontas incorporam processos complexos que podem ter desenvolvimentos posteriores. Para tal, essas soluções devem se comunicar com facilidade, segundo o conceito de interoperabilidade. Um sistema com características de interoperabilidade deve permitir acesso a suas funcionalidades segundo padrões de comunicação abertos, aceitos pelo mercado, de forma a tornar transparente, ou quase, o processo de integração. A principal estratégia de interoperabilidade atual para os sistemas computacionais é a criação de *web services*, dentro de uma arquitetura orientada para serviços. Essa estratégia foi utilizada na implantação de um *framework* de simulação para análise de custo de ciclo de vida, viabilizando que ferramentas como Flex, Delphi e Visual C# possam tirar proveito do núcleo de cálculo, criando interfaces próprias para problemas específicos dentro do domínio, além de relatórios personalizados.

Palavras-chave: Análise de custo de vida útil. Engenharia de sistemas. Simulação.

Abstract: *Computer systems have evolved a lot and several ready solutions incorporate complex processes that can be made available for further development. To this end these solutions must communicate easily, according to the concept of interoperability. A system with interoperability characteristics should allow access to its features according to open communication standards accepted by the market, in order to make transparent, or nearly so, the process of integration. The main current interoperability strategy for computer systems is the creation of Web Services within a service-oriented architecture. This strategy was used in the implementation of a simulation framework for Life Cycle Cost Analysis, enabling tools like Flex, Delphi and Visual C# to take advantage of the core calculation, creating own interfaces to specific problems within the domain, and custom reports.*

Keywords: *Life cycle cost analysis. Simulation. Systems Engineering.*

1 Introdução

Os primeiros sistemas informatizados apresentavam arquiteturas monolíticas, com soluções fechadas e grandes dificuldades em termos de comunicação. Devido a isso, vários componentes complexos de *software* deixavam de ser aproveitados em novos projetos, exigindo grande retrabalho e recorrência de esforços, muito presente em diversos projetos. Com a evolução das plataformas e a constatação de necessidades comuns em ambientes mais complexos, torna-se necessária uma nova abordagem focada em reuso e interoperabilidade.

Embora seja antigo o interesse da engenharia de sistemas pela interoperabilidade, inicialmente, ela só ocorria ao nível dos dados, com a definição de formatos e meios de armazenamento comuns. Posteriormente, os

[1] Laboratório de Redes e Arquitetura de Computadores e Sistemas Embarcados, Faculdade de Ciência da Computação, Fundação Centro Universitário Estadual da Zona Norte, Av. Manuel Caldeira Alvarenga, 1.203, CEP 23070-200, Campo Grande, Rio de Janeiro, RJ, Brasil
{deniscople@gmail.com}
[2] Programa de Pós-Graduação em Engenharia de Produção, Universidade Federal Fluminense, Rua Passo da Pátria, 156, sala 440, bloco E, CEP 24.020-140, Niterói, RJ, Brasil
{brick@producao.uff.br}

http://dx.doi.org/10.5335/rbca.2015.4660

sistemas se tornaram mais complexos, novos padrões de comunicação com o meio externo foram definidos, e a interoperabilidade evoluiu para o nível de processos e serviços. Para viabilizar a interoperabilidade, devem ser utilizados padrões de comunicação abertos, aceitos pelo mercado, e haver grande padronização das chamadas às diversas funcionalidades expostas pelos componentes de *software*, tornando os processos de integração bastante simplificados, ou até transparentes, para os desenvolvedores.

O interesse de empresas e órgãos governamentais pelo tema pode ser verificado pelo grande número de normas existentes, com o objetivo de trazer padronização aos elementos envolvidos nas mais diversas fases do ciclo de vida dos sistemas. Dentro desse enfoque, com o objetivo de viabilizar a interoperabilidade na área de simulação, o Departamento de Defesa dos Estados Unidos definiu um padrão arquitetural denominado *High Level Architecture* (HLA), que permite a integração de diversos simuladores e sistemas de apoio para a execução de complexas simulações distribuídas.

As arquiteturas de *software* da atualidade trabalham com foco na interoperabilidade de serviços, tendo como principal interface o uso de componentes do tipo *web service*. Segundo Klischewski e Askar [1], essa estratégia é utilizada por vários governos que estão preocupados com o compartilhamento de informações, de forma interoperável, nos sistemas de *eGovernment*, usualmente denominados G2G (*government-to-government*), em que, sob a perspectiva técnica, a adoção de arquiteturas orientadas a serviço é recomendada. Da mesma forma, a definição dos acessos aos processos de execução de simulações por intermédio de *web services* faz com que a integração em um ambiente HLA, também voltado para a interoperabilidade de serviços, torne-se uma tarefa mais simples.

Essa estratégia foi utilizada na implantação de um *framework* de simulação para análise de custo de ciclo de vida, viabilizando que ferramentas como Flex, Delphi e Visual C# possam tirar proveito do núcleo de cálculo, criando interfaces próprias para problemas específicos dentro do domínio, além de produzir relatórios personalizados. Um *framework* de simulação para o mesmo domínio de problema, capaz de assimilar alterações de modelos e dados, já havia sido desenvolvido por Cople e Brick [2], mas essa solução, embora apresentasse muitas características desejáveis de reuso, extensibilidade e evolubilidade, ficou muito distante dos ideais de interoperabilidade que essa nova solução, descrita por Cople e Brick [3], foi capaz de implantar.

Essa seção apresentou uma breve introdução ao problema que este trabalho procura resolver. A seção 2 aborda o conceito de interoperabilidade e suas principais características e aplicações, conforme tratado na literatura pesquisada. A seção 3 faz um breve resumo das tecnologias existentes para utilização de *web services*. Na seção 4, é descrito o domínio do problema para o qual o *framework* de simulação foi desenvolvido. O *framework* desenvolvido segundo a estratégia de *web services* é descrito na seção 5. Finalmente, na seção 6, são apresentadas as conclusões finais.

2 Interoperabilidade

Segundo Searle e Brennan [4], assumindo que o reuso é adequado e viável para a modelagem e simulação, pode-se observar que interoperabilidade e reutilização estão intimamente relacionadas. A fim de reutilizar um determinado componente em um sistema mais amplo, deve-se assegurar que ele possa interagir com os outros elementos do sistema (pelo menos aqueles com os quais deve trocar dados e informações).

Para Chen, Vallespir e Daclin [5], a interoperabilidade pode abranger quatro níveis em um sistema corporativo:

a) **interoperabilidade de dados**, referindo-se ao trabalho conjunto de diferentes modelos de dados (hierárquico, relacional, etc.) e das diferentes linguagens de consulta para encontrar e compartilhar informações provenientes de bases heterogêneas que, aliás, podem residir em diferentes máquinas com diferentes sistemas operacionais e diversos sistemas de gestão de bancos de dados;

b) **interoperabilidade dos serviços**, preocupada com a identificação, composição e funcionalidade conjunta de diversos serviços e aplicações (modelados e desenvolvidos independentemente), resolvendo as diferenças sintáticas e semânticas bem como as conexões para as diversas bases de dados heterogêneas;

c) **interoperabilidade dos processos**, destinando-se a fazer vários processos trabalharem de forma conjunta; geralmente em uma empresa, vários processos são executados em interações (em série ou

paralelo); no caso da rede corporativa, também é necessário estudar a forma de conexão entre processos internos de duas empresas para criar um processo comum;

d) **interoperabilidade de negócio**, referindo-se ao trabalho de uma forma harmonizada com os níveis da organização e da empresa, apesar de, por exemplo, os diferentes modos de tomada de decisão, métodos de trabalho, legislação, cultura da empresa e abordagens comerciais, etc., de modo que o negócio possa ser desenvolvido e compartilhado entre as empresas.

A interoperabilidade dos processos envolvidos no ciclo de vida útil de sistemas técnicos, desde a fase de concepção até a sua desativação final, tem sido o foco principal do desenvolvimento da engenharia de sistemas nos últimos anos. Evidentemente, uma condição necessária para viabilizar na prática a característica de interoperabilidade é a criação de normas e padrões. Muitas iniciativas com intensa participação de governos, empresas e organizações internacionais têm sido desenvolvidas visando ao que se convencionou chamar de engenharia de sistemas baseada em modelagem, que promove a integração de todos os processos envolvidos no ciclo de vida de um sistema técnico, ou seja, a integração de toda a cadeia produtiva envolvida no ciclo de vida de sistemas técnicos.

A atenção internacional ao problema de engenharia de sistemas complexos tem resultado na elaboração de normas e padrões, tratando-se dos seguintes temas:

a) **o que fazer?** (Interoperabilidade de serviços, processos e negócios);

b) **quão bem?** (Qualidade dos produtos e processos);

c) **com que dados?** (Interoperabilidade de dados);

d) **como modelar e representar os sistemas técnicos?** (Interoperabilidade de dados e modelos).

Muitas entidades estão envolvidas no processo de criação de normas aplicáveis à engenharia de sistemas complexos, entre as quais se destacam:

- **ISO** – International Organization for Standardization
- **IEC** – International Electrotechnical Commission
- **ANSI** – American National Standards Institute
- **INCOSE** – International Council for Systems Engineering
- **EIA** – Electronic Industry Association
- **DOD** – Department of Defense (USA)
- **OMG** – Object Management Group
- **IEEE** – Institute of Electrical and Electronic Engineers.

Normas como ANSI/EIA 632 [6], ISO/IEC/IEEE 15228 [7], ISO/IEC TR 24748-2 [8] e ISO/IEC 12207 [9] referem-se ao que fazer, prescrevendo o ciclo de vida de sistemas técnicos, quais processos devem ser desenvolvidos e o que deve ser feito em cada fase do ciclo de vida (foco nos processos da engenharia de sistemas). A norma ISO/IEC 33001 [10] preocupa-se com o quão bem foi feito, prescrevendo maneiras de avaliar a qualidade com que as instituições envolvidas estão desenvolvendo sistemas técnicos (foco na capacidade e qualidade para executar os processos da engenharia de sistemas).

Com relação à identificação dos dados, a norma ISO 10303 [11] prescreve maneiras de representar dados relativos a entidades e atributos do sistema e do processo de engenharia de sistemas para fins de automação e troca de informações entre programas, instituições e agentes envolvidos no processo (foco em informação). Finalmente, para a modelagem e representação dos sistemas técnicos, os padrões OMG SYSML [12] e DODAF [13] prescrevem maneiras de representar, por meio de símbolos e imagens, atributos e características do sistema técnico, abrangendo todas as visões, desde requisitos, passando por funções e arquitetura, até detalhamento de componentes (foco em linguagem).

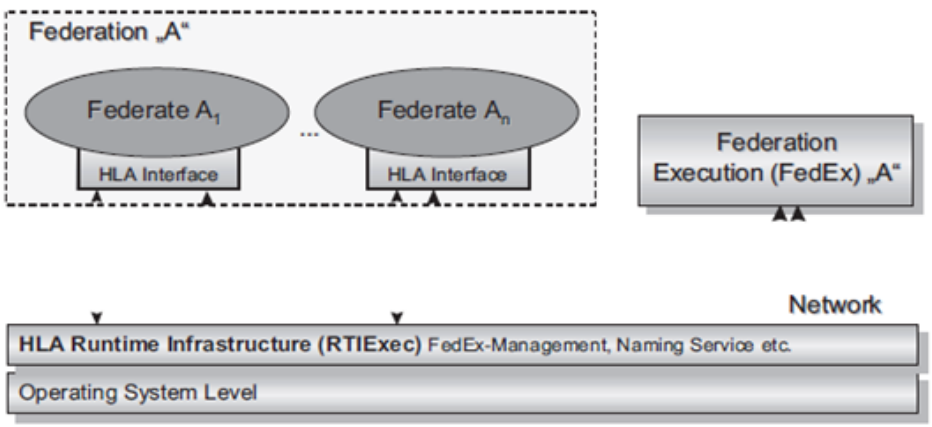
Há muito existe a preocupação em reutilizar os sistemas existentes na construção de sistemas maiores e mais complexos por meio da interoperabilidade, o que levou à definição de arquiteturas e metodologias que permitissem essa integração. Um exemplo disso foi o surgimento da *High Level Architecture* voltada para a

integração de módulos de simulação que podem cooperar para um objetivo comum, executando-o de forma distribuída. A definição da HLA foi iniciada pelo Departamento de Defesa dos Estados Unidos, em 1995, e a responsabilidade de sua evolução delegada ao IEEE's Simulation Interoperability Standards Committee no ano de 1997. A partir disso, a HLA passou a ser regulamentada pela norma IEEE 1516, *Standard for Modeling and Simulation High Level Architecture – Framework and Rules*, cuja última revisão é do ano de 2010 [14].

Segundo Morse et al. [15], a HLA provê uma metodologia comum para a modelagem de sistemas de simulação distribuídos, conectando simulações e interfaces com sistemas interativos, segundo uma abordagem contemporânea em que o modelo de dados e a semântica da arquitetura são separados das funções e métodos para intercâmbio de informações.

Strassburger [16] define de forma resumida a arquitetura HLA, citando que simulações individuais e outros participantes de uma simulação distribuída são denominados *federates*, enquanto um conjunto de *federates* que supostamente atuam de forma cooperativa, obedecendo a certas padronizações e a um modelo de objetos definido, forma o que é chamado de *federation*. Ainda segundo o autor, os *federates* utilizam a *Runtime Infrastructure* (RTI) para se comunicar, enquanto o HLA define um caminho bidirecional entre os *federates* e o RTI, como pode ser observado na Figura 1.

Figura 1: Visão funcional de uma simulação distribuída sob HLA



Fonte: Strassburger, 2006 [16].

Segundo Papazoglou et al. [17], *Service Oriented Computing* (SOC) é um paradigma que define o uso de serviços para o desenvolvimento de aplicações distribuídas a baixo custo, de forma rápida, interoperável e evolutiva. Esse paradigma é suportado pelas arquiteturas orientadas ao serviço (SOA), cuja interface padrão para disponibilização e integração de serviços é realizada por intermédio de *web services*. De acordo com Castagna, Gesbert e Padovani [18], os *web services* são componentes distribuídos com os quais clientes podem se conectar e se comunicar por meio de protocolos de comunicação padrão e mensagens em formato independente de plataforma. Outros autores, como Jacobi e Radul [19], Mehta, Kanungo e Chandwani [20], Tsai et al. [21], defendem a abordagem do uso de *web services* para a construção de sistemas e arquiteturas com processamento distribuído.

Para Dan, Johnson e Carrato [22], a reutilização de serviços em um ambiente SOA proporciona muitos benefícios: melhora a agilidade na implementação de soluções por meio de uma rápida montagem de novos processos de negócios, a partir de serviços existentes, para atender às mudanças das necessidades de mercado, reduz os custos, não apenas evitando duplicação de código ao reutilizar funções de negócios semelhantes ao longo dos múltiplos processos, mas também durante todo o ciclo de vida do sistema, abrangendo a implantação de serviços e gestão, diminui os riscos com o reuso de código e em um ambiente de execução bem testados. Essa linha, em particular a implantação de simulação por eventos discretos (*discrete event simulation* – DEVS) sob SOA, com o objetivo de distribuir o processamento, é defendida por autores como D'Ambrogio et al. [23], Seo e Zeigler [24], Mittal, Risco-Martín e Zeigler [25].

A utilização de *web services* não apenas permite o processamento distribuído, como também aumenta a interoperabilidade ao utilizar um padrão de mercado reconhecido, o que faz com que ferramentas especializadas

possam ser utilizadas para a implantação de cada aspecto de um sistema complexo. Segundo Fan, Zhang e Fan [26], a multidisciplinaridade colaborativa das tecnologias de simulação é amplamente utilizada na definição e construção de produtos complexos, sendo a combinação de HLA e *web services* um meio efetivo para a obtenção de integração e interoperabilidade nas simulações distribuídas heterogêneas.

3 Tecnologias para *web services*

Como toda arquitetura de componentes que visa apresentar interoperabilidade e capacidade de computação distribuída, os *web services* também apresentam um padrão de definição de interfaces e um serviço de registro e localização. De acordo com Erl [27], a plataforma original de tecnologia de *web services* é composta das principais especificações e tecnologias abertas a seguir: *Web Services Description Language* (WSDL), XML *Schema Definition Language* (XSD), *Simple Object Access Protocol* (SOAP) e *Universal Description, Discovery and Integration* (UDDI).

O que se pode observar nesses padrões é que o XML é utilizado em todos os níveis da arquitetura. No entanto, o XML sem um vocabulário é meramente uma sintaxe aberta, sem as formalizações necessárias à definição de um protocolo de comunicação e troca de dados. Logo, para aplicativos reais, um vocabulário deve ser definido, e as duas tecnologias disponíveis para tal são o *Document Type Definition* (DTD) e XML *Schema Definition Language* (XSD). Qualquer serviço criado deve ser registrado via UDDI, passando a fornecer a interface de utilização desse serviço, envolvendo o formato das mensagens de chamada e de retorno, por meio de um arquivo WSDL. Qualquer mensagem transitada entre o servidor e o usuário do serviço deve ser envelopada segundo o formato do SOAP.

Dentro desse ambiente, os arquivos XSD têm o papel de definir estruturas complexas a serem enviadas ou recebidas pelos serviços, sendo esses tipos utilizados na mensagem SOAP para determinar o domínio de determinados atributos.

O processo completo para a definição e uso de *web services* envolve os seguintes passos:

1. criação do serviço e registro no servidor via UDDI;
2. localização do serviço pelo cliente por meio do catálogo UDDI;
3. recuperação pelo cliente do formato de chamada e retorno por intermédio do WSDL e de possíveis arquivos XSD fornecidos;
4. preparação pelo cliente do envelope SOAP de chamada e retorno a partir das informações obtidas no passo anterior;
5. envio da mensagem SOAP de chamada pelo cliente;
6. execução do *web service* no servidor;
7. recepção pelo cliente da mensagem SOAP de retorno.

A Tabela1 apresenta um paralelo entre os elementos funcionais de *web services* e aqueles encontrados em outras tecnologias.

Tabela 1: Paralelo entre *web services* e outras tecnologias.

	CORBA	RMI	JEE	*Web services*
Registro e localização	COS *Naming*	JNDI	JNDI	UDDI
Descritor de interface	IDL	Java	Java	WSDL
Protocolo	IIOP	RMI	RMI-IIOP	SOAP

Fonte: elaboração dos autores.

O Java permite a criação de *web services* com o uso de anotações. Durante a implantação do serviço programado na linguagem, as anotações são reconhecidas pelo servidor *web* e o descritor de interface WSDL é gerado automaticamente.

Um exemplo de *web service* criado com o uso de anotações da tecnologia JEE5 pode ser observado na Figura 2.

Figura 2: Exemplo de código Java anotado para criação de *web services*

```
@WebService()
public class AluguelWS {
    @EJB
    private AluguelFacadeRemote ejbRef;

    @WebMethod(operationName = "create")
    @Oneway
    public void create(@WebParam(name = "aluguel")
    Aluguel aluguel) {
        ejbRef.create(aluguel);
    }

}
```

Fonte: elaboração dos autores.

De acordo com Chen [28], SOC, incluindo SOA, *web services* e seus modelos de dados e processamento, representam um novo paradigma computacional que muda a forma para o desenvolvimento de sistemas e uso de *software*.

Nos ambientes computacionais voltados para a integração de serviços, não importa a linguagem de programação que realiza cada componente, pois a interface comum permite que cada serviço funcione como uma caixa-preta, expondo apenas os formatos de entrada e saída de dados por meio dessa interface para qualquer tecnologia compatível. A partir disso, os *softwares* são implantados como clientes de sequências de acessos a diferentes serviços, segundo um *workflow* determinado pelas necessidades e lógica do sistema.

Segundo Erl [27], os serviços são sistemas fisicamente independentes, com características de projeto distintas que dão suporte à obtenção dos objetivos estratégicos associados à computação orientada a serviços, sendo que cada serviço recebe seu próprio contexto funcional distinto e tem um conjunto de capacidades relacionadas a esse contexto, sendo as capacidades adequadas a invocações externas comumente expressas via contrato público. Para Chen [28], o relacionamento entre os componentes constituintes no SOA é baseado em troca de serviços, que retornam os dados solicitados em vez de importar o código que processa os dados. O autor ainda defende a interoperabilidade ao citar que essa arquitetura fracamente acoplada torna o desenvolvimento independente de plataforma possível.

Esse ambiente satisfaz às necessidades de interoperabilidade dos heterogêneos sistemas de simulação computacional atuais, levando a um novo paradigma em termos de simulação, denominado *Service-Oriented Modelling and Simulation Framework* (SOMSF), que, segundo Chen [28], ao se basear em SOC, podem ser construídos *frameworks* e ambientes de desenvolvimento que suportam simulações reconfiguráveis dinamicamente, além de diferentes composições de serviços de simulação.

Essa nova interpretação com relação ao foco de desenvolvimento permite também a integração de sistemas de simulação antigos com novos componentes, permitindo que ocorra grande reuso e que a construção de novos sistemas de simulação esteja completa em um tempo muito menor.

Erl [27] enumera certas vantagens com relação ao uso de SOA:

a) as soluções podem ser construídas eficientemente porque só precisam se preocupar em atender um conjunto restrito de requisitos associados a um conjunto limitado de processos de negócio;
b) o esforço de análise é simples e direto, a partir do momento em que os analistas se concentram em apenas um processo por vez e, desse modo, só se preocupam com as entidades e os domínios associados a este único processo;
c) a criação de soluções segue um ponto de vista tático, satisfazendo ao único propósito de automatizar um conjunto específico de processos de negócio, e esse escopo funcional predefinido simplifica não só o desenvolvimento da solução, como a arquitetura de aplicativos subjacentes;
d) o ciclo de vida de desenvolvimento de cada solução é simplificado e relativamente previsível;
e) organizações que constroem repetidamente aplicações descartáveis podem se beneficiar das últimas inovações tecnológicas a cada novo projeto.

Baseando-se em *web services* como interfaces de comunicação com os clientes, ao mesmo tempo em que facilita o desenvolvimento de soluções, qualquer servidor SOA apresenta grande complexidade em termos de sua organização interna, justificada pela necessidade de integração de novas tecnologias e sistemas legados, dentro de uma análise de processos, ou serviços, que trabalham de forma conjunta, independente da origem de cada um.

Serviços também podem ser combinados no servidor, segundo uma metodologia denominada orquestração, em que é definida a sequência e forma de chamadas de serviços secundários, acessados por um serviço principal que faz a intermediação com o cliente. Para organizar essas chamadas, é utilizada uma linguagem denominada *Business Process Execution Language* (BPEL).

Para Erl [27], existem alguns pontos principais no que se refere à composição de serviços no SOA:

a) a composição de serviços associa-se estreitamente à capacidade de reuso de serviços, já que a composição pode ser vista como uma forma de reuso;
b) é necessário assegurar que os serviços sejam projetados para participar como membros eficazes de diversas composições, mesmo que não existam requisitos imediatos de composição;
c) a aplicação desse princípio pode afetar praticamente todas as partes de uma arquitetura focada em serviços.

Outro tipo de *web service* bastante utilizado no mercado é o Representational State Transfer (RESTful), que trabalha com a representação do estado de objetos, não sendo direcionado especificamente para a exposição de operações, como ocorre no SOAP. Alguns autores, como Wagh e Thool [29], definem as arquiteturas baseadas em *web services* REST como *Resource Oriented Architecture* (ROA), já que todo o formalismo e ferramental do SOA trabalha melhor com o padrão SOAP.

4 Domínio do problema de análise de custo de ciclo de vida

A importância da análise de custo de ciclo de vida (ACCV) é ressaltada em muitos trabalhos, como os de Rathod, Rohan e Puranik [30], Vinodh e Rathod [31], Korpi e Ala-Risku [32], Xie e Simon [33], bem como o uso da simulação na resolução desse tipo de problema. Cople e Brick [3] descrevem com detalhes o domínio do problema para ACCV. Assim, nesta seção será feita apenas uma descrição genérica, em que as seguintes premissas gerais são assumidas para a ACCV:

a) uma organização utiliza ativos físicos (sistemas técnicos tais como viaturas, embarcações, aeronaves, máquinas, sistemas de comunicações e computação, etc.) em seus processos produtivos;
b) os processos produtivos são executados no âmbito de unidades operacionais (órgãos) que podem estar distribuídos geograficamente. O regime de funcionamento desses órgãos pode ser definido arbitrariamente;
c) para garantir a disponibilidade desses ativos, ela cria uma estrutura de manutenção e abastecimento, que também pode estar distribuída geograficamente. Essa estrutura pode estar organizada em escalões, ou níveis, de manutenção, em função das capacidades existentes para efetuar ações de manutenção em cada caso;
d) os ativos podem ser desmembrados em partes componentes, que podem ser trocadas e eventualmente reparadas quando apresentarem defeitos;
e) uma vez detectado um defeito em uma parte de um ativo, essa parte pode ser reparada no próprio local ou enviada para reparo em outro órgão de manutenção;
f) estoques de partes sobressalentes são armazenados em órgãos de abastecimento. Partes reparáveis podem retornar ao estoque, após sofrerem ações;
g) todas as atividades relacionadas aos ativos físicos, ou a suas partes componentes, utilizam recursos que estão associados a modelos de custos. Essas atividades são executadas ao longo do ciclo de vida (desde a concepção dos ativos até sua desativação e descarte) em instantes gerados por modelos de geração de eventos. Esses eventos podem ser definidos a priori ou serem dependentes de estados do sistema durante a simulação.

O modelo para ACCV é muito simples. Ele apenas computa o somatório dos custos de todas as atividades relacionadas a ativos físicos e/ou suas partes componentes, desde sua concepção até sua desativação final. Esses custos, que ocorrem ao longo de períodos de tempo muito extensos (da ordem de dez a trinta anos), podem sofrer ajustes, usando taxas de desconto e técnicas de valor presente.

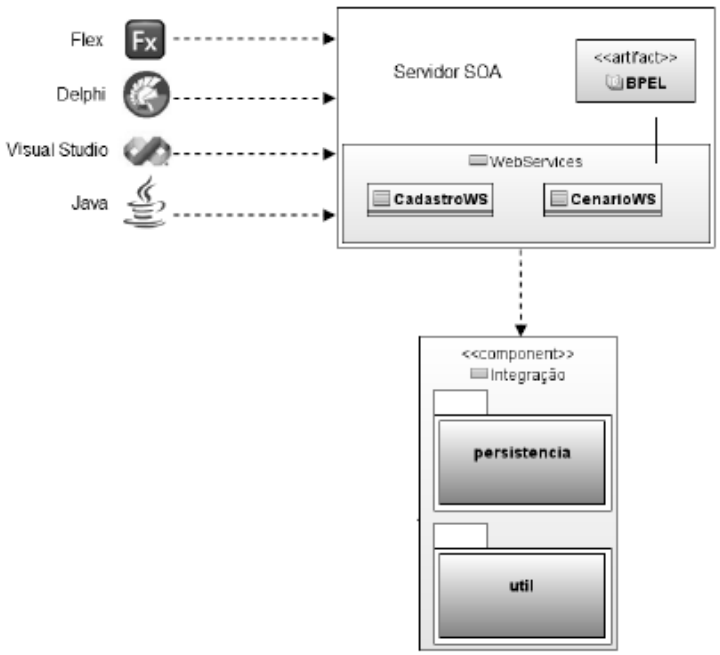
5 *Framework* de ACCV baseado em *web services*

Na arquitetura do *framework* construído o uso de *web services* dentro de uma arquitetura SOA divide a funcionalidade do *software* em cinco serviços distintos, disponibilizados para o desenvolvedor:

a. intermediação de dados cadastrais;
b. construção de cenários;
c. execução de simulações;
d. fornecimento de resultados;
e. geração de código-fonte.

O serviço de intermediação de dados cadastrais permite a qualquer desenvolvedor a criação de interfaces visuais para cadastro de dados de catálogo genéricos ou focados em problemas específicos da organização em que trabalha, com independência de linguagem, garantindo a interoperabilidade com ferramentas de criação como Flex, Delphi, Visual C# ou qualquer outra com suporte a *web services*. A construção de cenários segue o mesmo princípio, permitindo a criação de modelos de cenários parametrizados específicos para os objetivos de análise em cada situação. Os elementos arquiteturais associados aos serviços de intermediação de dados e construção de cenários podem ser observados na Figura 3.

Figura 3: Serviços de intermediação de dados e construção de cenários



Fonte: elaboração dos autores.

As simulações do *framework* ocorrem com a utilização de paralelismo, em que é implantado o conceito de grupo de simulações para cada cenário, podendo-se observar o comportamento de uma simulação individual do grupo em termos analíticos, com o acompanhamento dos eventos ponto a ponto, ou de forma sintética, com a análise dos elementos financeiros aferidos com base em uma estrutura de desdobramento de custos. Da mesma forma, a análise sintética pode ser efetuada sobre o conjunto de simulações completo, obtendo uma média comportamental que deverá se aproximar muito mais da situação real.

A arquitetura para extração de resultados das simulações é muito similar à do primeiro módulo, voltado para intermediação de dados, sendo a interface externa baseada em dois serviços:

a) sintéticoWS, que fornece dados sintetizados para grupos de simulações;
b) analíticoWS, que fornece dados sobre simulações individuais.

O uso de várias simulações simultâneas, com o objetivo de atingir um resultado agregado próximo da realidade, segue a estratégia de simulação de Monte Carlo.

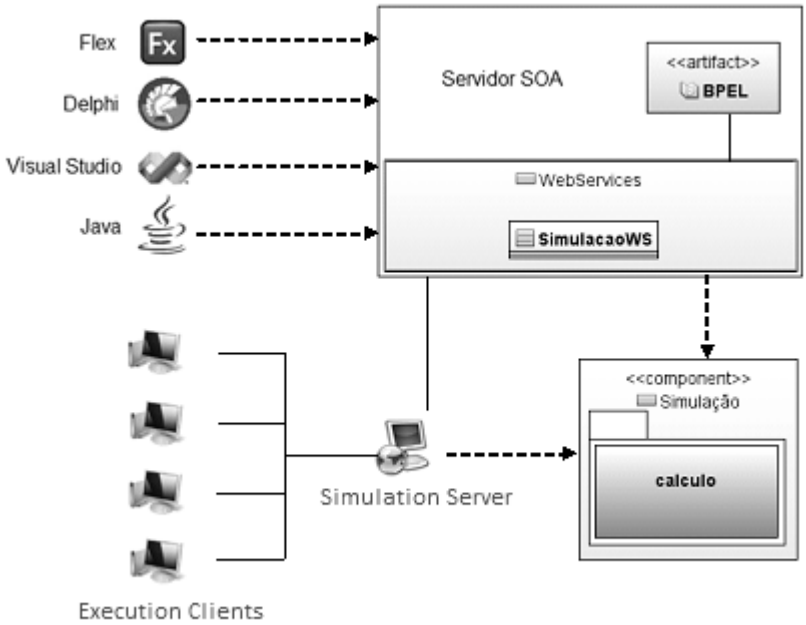
Para satisfazer ao modelo de execução do *framework*, com um grande número de simulações sendo executado, seria inviável a utilização de apenas uma máquina processando as simulações. Foi disponibilizado um serviço para solicitação de execução de grupos de simulações, que também provê acompanhamento do percentual de conclusão das solicitações.

Mesmo internamente, o uso de *web services* demonstrou-se interessante na comunicação com o ambiente de execução distribuída, apoiado em três serviços baseados em *Sockets*:

a) porta 2525: Gerador de Solicitações de Simulação (GSS);
b) porta 2526: Coordenador de Simulações em Execução (CSE);
c) porta 2527: Receptor de Resultados Simulados (RRS).

A arquitetura de fornecimento do serviço pode ser observada na Figura 4, que apresenta também a arquitetura interna de execução de simulações distribuídas.

Figura 4: Arquiteturas de fornecimento de serviços e execução de simulações

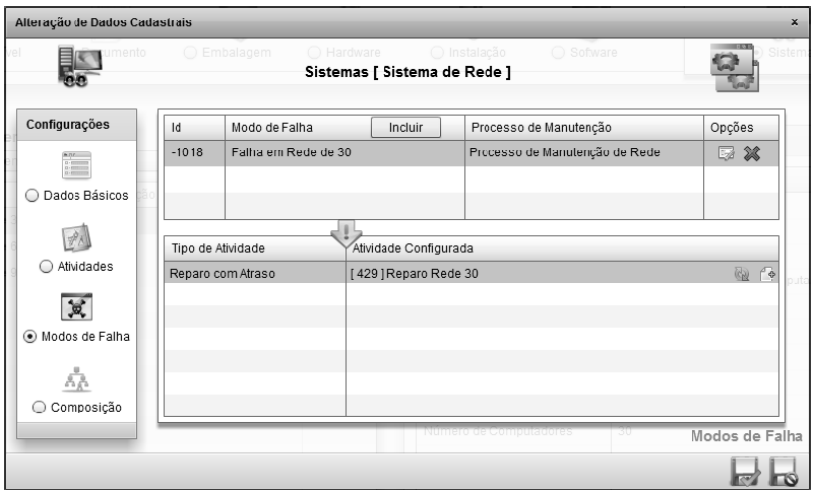


Fonte: elaboração dos autores.

As classes internas do *framework* de ACCV são anotadas, o que permite a extensão direta em Java com pouco esforço de desenvolvimento. Essas anotações constituem o mecanismo necessário para que o *framework* identifique qualquer necessidade de adição de entradas na base de dados, representativa dos metadados dos componentes de simulação adicionados. Essa geração de código também é apoiada por um *web service*, que recebe o nome do novo componente de simulação e seu código-fonte, delegando para o núcleo do *framework* a tarefa de integrar o componente recebido via SOAP.

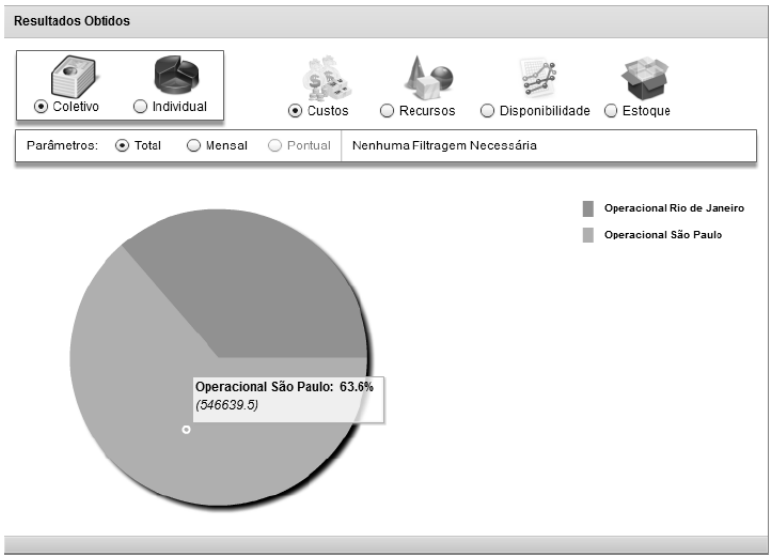
Como exemplo de utilização, é apresentada na Figura 5 a tela de um sistema em Flex que efetua o cadastro de elementos de simulação com o acesso aos *web services* do *framework*, bem como o resultado de uma simulação nesse sistema, como apresentado na Figura 6, obtido segundo a mesma metodologia de interoperabilidade.

Figura 5: Exemplo de sistema Flex que acessa o *framework*



Fonte: elaboração dos autores.

Figura 6: Resultado parcial de simulação efetuada no *framework* e exibido por meio de Flex



Fonte: elaboração dos autores.

6 Conclusões

Com a plena interoperabilidade obtida com o uso de *web services* no desenvolvimento de um *framework* de simulação para análise de custo de ciclo de vida de sistemas técnicos, foi possível aproveitar as melhores características da linguagem Java em termos de processamento e distribuição, ao mesmo tempo em que telas muito complexas e com grande requinte visual foram construídas por meio do Flex. Novos sistemas poderão ser construídos em ambientes como Visual Studio .Net, PHP ou qualquer outra linguagem, devido às características de comunicação e exposição de interface dos *web services*.

Isso também deixa o simulador alinhado com as tendências atuais para a criação de simulações distribuídas, já que o modelo de utilização do *framework* é baseado em serviços, com exposição por meio de um padrão amplamente utilizado, o que viabiliza a integração com outros simuladores e ferramentas de interesse em ambiente HLA.

Quanto à divisão em áreas de serviços especializadas, tornou-se possível criar novas ferramentas focando diferentes utilizações do *framework*, como criação de cenários e definição de modelos de cálculo, por meio do simples uso do SOAP.

Finalmente, a divisão do processamento por meio da rede com o uso de *Sockets*, visando aproveitar o poder computacional das diversas máquinas disponíveis, ampliou muito as dimensões de problemas que podem ser considerados viáveis para resolução por meio de *framework*. O uso de *web services* para intermediar o acesso a esses serviços baseados em *Socket* viabiliza o aproveitamento do sistema de cálculo pelas mais diversas ferramentas, trazendo também para esse nível o conceito de interoperabilidade.

Referências

[1] KLISCHEWSKI, R.; ASKAR, E. Linking service development methods to interoperability governance: The case of Egypt. *Government Information Quarterly*, v. 29, , p. S22–S31, Jan. 2012. Supl. 1.

[2] COPLE, D. G.; BRICK, E. S. Um sistema de simulação para análise de custo de vida útil de sistemas técnicos. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 37, 2005, Gramado, RS. *Anais...* Rio de Janeiro: Sobrapo, 2005. p. 2295-2306.

[3] COPLE, D. G.; BRICK, E. S. A simulation framework for technical life cycle cost analysis. *Simulation Modelling Practice and Theory*, Amsterdam, v. 18, n. 1, p. 9-34, Jan. 2010.

[4] SEARLE, J.; BRENNAN, J. General Interoperability Concepts. França: RTO, Report apresentado em conferência da NATO, Integration of Modelling and Simulation, Educational Notes RTO-EN-MSG-067, Paper 3, 2007. p. 3.1-3.8.

[5] CHEN, D.; VALLESPIR, B.; DACLIN, N. An Approach for Enterprise Interoperability Measurement. França: MoDISE-EUS'2008, Model Driven Information System Engineering-Enterprise, User and System Models, 2008. p. 1-12.

[6] ANSI/EIA. ANSI/EIA 632 – Processes for Engineering a System. Philadelphia, PA, USA: ANSI – American National Standards Institute/EIA – Electronic Industries Association, 2003.

[7] ISO/IEC/IEEE. ISO/IEC/IEEE 15288:2015. Systems and Software Engineering – System Life Cycle Processes. Geneva, SWZ: ISO – International Organization for Standardization/IEC – International Electrotechnical Commissions/IEEE – Institute of Electrical and Electronics Engineers, 2015.

[8] ISO/IEC. ISO/IEC TR 24748-2:2011 Systems and software engineering – Life cycle management – Part 2: Guide to the application of ISO/IEC 15288 (System life cycle processes). Geneva, SWZ: ISO – International Organization for Standardization/IEC – International Electrotechnical Commission, 2011. p. 76.

[9] ISO/IEC. ISO/IEC 12207:2008 Systems and software engineering – Software life cycle processes. Geneva, SWZ: ISO – International Organization for Standardization/IEC – International Electrotechnical Commission, 2008. p. 123.

[10] ISO/IEC. ISO/IEC 33001:2015 Information technology – Process assessment – Concepts and terminology. Geneva, SWZ: ISO - International Organization for Standardization/IEC – International Electrotechnical Commission, 2015. p. 76.

[11] ISO. ISO 10303-233:2012 Industrial automation systems and integration – Product data representation and exchange – Part 233: Application protocol: Systems engineering. Geneva, SWZ: ISO – International Organization for Standardization, 2012. p. 800.

[12] OMG. OMG Systems Modeling Language (OMG SysML™) Version 1.4. [S. l.]: OMG – Object Management Group, 2015. p. 346.

[13] DOD. DoDAF Architecture Framework Version 2.02. Washington, DC, USA: DOD – Department of Defense, ago. 2010. p. 289.

[14] IEEE. IEEE 1516:2010 Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Framework and Rules. [S. l.]: IEEE – Institute of Electric and Electronics Engineers, 2010.

[15] MORSE, K. L. et al. Enabling Simulation Interoperability. *Computer*, Los Alamitos, v. 39, n. 1, p. 115-117, 2006.

[16] STRASSBURGER, S. Overview about the High Level Architecture for Modelling and Simulation and Recent Developments. *Simulation News Europe*, Vienna, v. 16, n. 2, p. 5-14, 2006.

[17] PAPAZOGLOU, M. et al. Service-Oriented Computing: state of the art and research challenges. *Computer*, Los Alamitos, v. 40, n. 11, p. 38-45, 2007.

[18] CASTAGNA, G.; GESBERT N.; PADOVANI, L. A theory of Contracts for Web Services. *ACM Transactions on Programming Languages and Systems*, New York, v. 31, n. 5, p. 19.1-19.61, 2009.

[19] JACOBI, I.; RADUL, A. A RESTful Messaging System for Asynchronous Distributed Processing. In: INTERNATIONAL WORKSHOP ON RESTFUL DESIGN, 1, 2010, New York. *Proceedings...* New York, USA: ACM, 2010. p. 45-53.

[20] MEHTA, H.; KANUNGO, P.; CHANDWANI, M. Generic data access and integration service for distributed computing environment. *International Journal of Grid Computing & Applications (IJGCA)*, Australia, v. 1, n. 1, p. 14-21, 2010.

[21] TSAI, W. et al. A service-oriented modeling and simulation framework for rapid development of distributed applications. *Simulation Modelling Practice and Theory*, Amsterdam, v. 14, n. 6, p. 725-739, 2006.

[22] DAN, A.; JOHNSON, R. D.; CARRATO, T. SOA Service Reuse by Design. In: INTERNATIONAL WORKSHOP ON SYSTEMS DEVELOPMENT IN SOA ENVIRONMENTS, 2, 2008, Leipzig, Germany. *Proceedings...* New York, USA: ACM, 2008. p. 25-28.

[23] D'AMBROGIO, A. et al. A MDA-based approach for the development of DEVS/SOA simulations. In: SPRING SIMULATION MULTICONFERENCE, 10, 2010, Orlando, Florida, USA. *Proceedings...* San Diego, CA, USA: Society for Computer Simulation International, 2010, article 142.

[24] SEO, C.; ZEIGLER, B. P. DEVS namespace for interoperable DEVS/SOA. In: WINTER SIMULATION CONFERENCE, 2009, Austin, TX, USA. *Proceedings...*, USA: IEEE, 2009. p. 1311-1322.

[25] MITTAL, S.; RISCO-MARTÍN, J. L.; ZEIGLER, B. P. DEVSML: Automating DEVS Execution over SOA Towards Transparent Simulators. In: SPRING SIMULATION MULTICONFERENCE, 2007, Norfolk, Virginia, USA. *Proceedings...* San Diego, CA, USA: Society for Computer Simulation International, 2007. Vol. 2, p. 287-297.

[26] FAN, M.; ZHANG, J.; FAN, Y. A heterogeneous model integration and interoperation approach in distributed collaborative simulation environment. *International Journal of Internet Manufacturing and Services*, Genebra, v. 2, p. 294-309, 2010.

[27] ERL, T. *SOA*: princípios de *design* de serviços. Tradução: Carlos Schafranski e Edson Furmankiewicz. São Paulo: Pearson Brasil, 2009.

[28] CHEN, Y. Modeling and Simulation for and in Service-Oriented Computing paradigm. *ACM, Simulation Transactions*, USA, v. 83, n. 1, p. 3-6, 2007.

[29] WAGH, K.; THOOL, R. A comparative study of soap Vs rest Web Services provisioning techniques for mobile host. *Journal of Information Engineering and Applications*, v. 2, n. 5, p. 12-16, 2012.

[30] RATHOD, G.; ROHAN; PURANIK, V. S. Life cycle assessment and simulation: enablers of sustainable product design. *International Journal of Research in Engineering and Technology*, Bangalore, v. 3, p. 851-855, 2014. Special Issue 3.

[31] VINODH, S.; RATHOD, G. Application of life cycle assessment and Monte Carlo simulation for enabling sustainable product design. *Journal of Engineering, Design and Technology*, v. 12, n. 3, p. 307-315, 2014.

[32] KORPI, E.; ALA-RISKU, T. Life cycle costing: a review of published case studies. *Managerial Auditing Journal*, v. 23, n. 3, p. 240-261, 2008.

[33] XIE, X.; SIMON, M. Simulation for product life cycle management. *Journal of Manufacturing Technology Management*, v. 17, n. 4, p. 486-495, 2006.