XIX Seminário Nacional de Distribuição de Energia Elétrica

SENDI 2010 – 22 a 26 de novembro

São Paulo - SP - Brasil

Integração de Sistemas Corporativos adequada às redes inteligentes (*Smart grids*)

Ezequiel C. Pereira	Lucas Barros H. Silva
CEMIG Distribuição SA	Axxiom Soluções Tecnológicas SA
ezequiel.pereira@cemig.com.br	lucas.silva@cemig.com.br

PALAVRAS-CHAVE

Arquitetura orientada a serviços – SOA, Barramento de serviços corporativos – ESB, Integração de sistemas corporativos – EAI, Modelo de informações comum - CIM, Redes Inteligentes - S*mart Grids*.

RESUMO

Este trabalho descreve uma experiência da CEMIG Distribuição SA na criação de um projeto piloto de uma arquitetura de software para a corporação adequada às demandas das redes inteligentes, *Smart Grids*. Neste contexto, um dos desafios é a integração das informações dos elementos de rede elétrica, sistemas de engenharia e outros sistemas de informação da concessionária, não só para a troca de informações internas, mas também para a interação com outros agentes (e.g.: ANEEL, ONS).

O projeto piloto utilizou conceitos de arquitetura orientada a serviços – SOA, modelo comum de informações – CIM e barramento de serviços corporativos – ESB para realizar uma de integração de sistemas corporativos – EAI. Serão relatados os conceitos e as tecnologias utilizadas, o desenho arquitetônico empregado, as dificuldades encontradas no desenvolvimento, os resultados alcançados e os benefícios da adoção de uma integração orientada a serviços e baseada em modelos, como o CIM.

A arquitetura do projeto mostrou-se viável tecnicamente e capaz de suportar as demandas das tecnologias futuras como as redes inteligentes (S*mart Grids*).

1. INTRODUÇÃO

Nos últimos anos, os governos e a indústria têm investido no desenvolvimento das redes inteligentes (*Smart Grids*). Um dos grandes motivadores para essa iniciativa foi o apagão de dois dias ocorrido nos EUA em 2003 que atingiu cerca de 50 milhões de pessoas e cujos prejuízos foram de US$ 6 bilhões. Mais recentemente, a crise financeira internacional de 2008 incentivou ainda mais as iniciativas *Smart Grid* através de financiamentos governamentais, com destaque ao EUA (GARTNER, 2009b).

O advento das redes inteligentes, *Smart Grids*, promoverá uma completa mudança no modelo do negócio do setor elétrico, alterando significantemente a relação entre a concessionária e o consumidor. Dos primeiros passos até as idéias mais avançadas, a rede inteligente exigirá grandes investimentos principalmente em sistemas computacionais corporativos, telecomunicações e equipamentos de distribuição de energia elétrica.

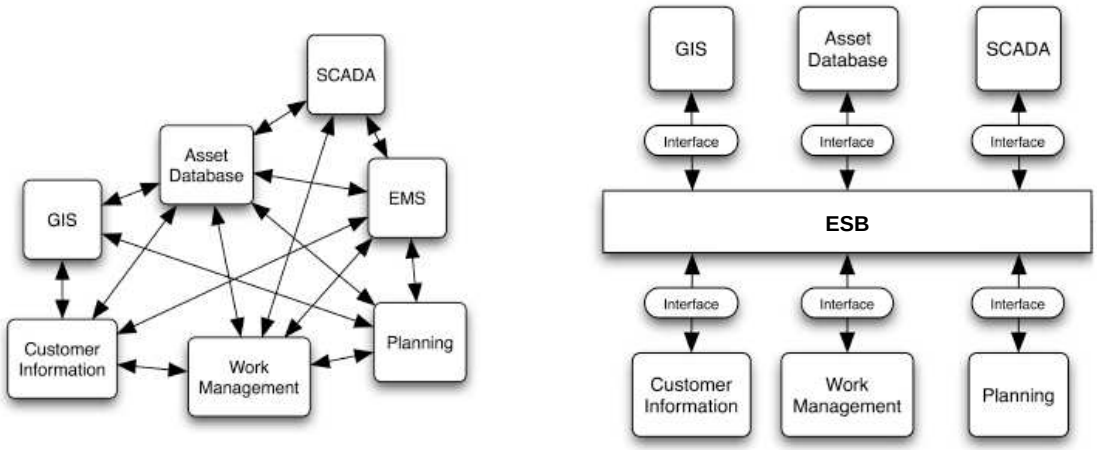
As redes inteligentes proporcionarão o fornecimento de energia com melhor qualidade e confiabilidade, além de novos serviços para os consumidores como: planos de tarifas baseadas em

perfil de consumo (*real-time pricing* - RVP) e gerenciamento de carga. Na distribuição de energia elétrica, as redes inteligentes possibilitarão funções como: Localização de Falha, Isolamento e Restabelecimento do Sistema (FLISR), Despacho de Potência Reativa (*Var Disptach*- VD), Controle de Tensão (*Voltage Control*-VC) e Gerenciamento de Carga do Transformador (*Transformer Load Managemen*-TLM).

Um dos critérios chaves para a implantação da arquitetura das redes inteligentes é o tratamento dos sistemas legados da concessionária, com o desenvolvimento de planos de migração desses sistemas (EPRI, 2003), escopo deste trabalho.

As concessionárias de energia elétrica possuem um grande número de sistemas computacionais, tais como o sistema de informações geográficas – GIS, sistemas supervisórios – SCADA, sistema de relacionamento com o consumidor – CRM entre outros. Tradicionalmente, as trocas de informações entre estes sistemas são realizadas por conversores/adaptadores específicos, ponto-a-ponto, o que resulta em um alto custo de desenvolvimento e manutenção, visto que a inclusão e/ou substituição de um sistema, implica na implementação de novos conversores de dados.

O padrão atualmente proposto (EPRI, 2009) é o uso de um barramento de serviços corporativos - ESB (*Enterprise Service Bus*) que substitui os conversores específicos entre as aplicações por um único conversor/interface entre a aplicação e o barramento, permitindo a troca de mensagens entre todas as aplicações sem os inconvenientes dos conversores específicos.



Integração ponto-a-ponto Integração através de barramento de serviços

Figura 1 – Comparativos entre integrações de sistemas (MCMORRAN, 2007)

Para possibilitar a interoperabilidade e integração dos sistemas foi publicado um padrão de modelo de informação nas séries das normas IEC 61970 e IEC 61968. O CIM (*Commom information model*) descreve a maior parte dos componentes do sistema elétrico, e é atualmente proposto (GARTNER, 2006) para integração de sistemas das concessionárias de energia elétrica.

O projeto piloto desenvolvido na Cemig Distribuição S.A utilizou uma arquitetura orientada à serviços (*Service Oriented Architecture* – SOA) e o modelo CIM para a integração dos sistemas, visando a adaptação às tecnologias futuras das redes inteligentes (S*mart Grids*).

Um dos objetivos do projeto foi a avaliação das tecnologias disponíveis no mercado e a criação de um piloto envolvendo alguns dos sistemas da concessionária, como o sistema de informação geográfica, sistema de gerenciamento de rede e o sistema de serviços de campo. O conceito de portal de quarta geração também foi abordado, com a disponibilização de informações oriundas dos diversos sistemas, dispostas de uma maneira integrada.

Adotando o CIM, além de permitir uma integração mais flexível, escalável e ágil para as mudanças nos processos do negócio, proporciona também um controle da exposição das informações da concessionária (GARTNER, 2006). Os ganhos para a concessionária vão desde o gerenciamento de informações empresariais - EIM (*Enterprise Information Management*), a uma melhor eficiência operacional e benefícios em analise avançada de ativos EAM (*Enterprise Asset Management*) (GARTNER, 2009a).

2. DESENVOLVIMENTO

Nessa seção será detalhado o modelo de informação comum CIM (sub-item 2.1), as interfaces genéricas para troca de informações (sub-item 2.2), a arquitetura de referência adotada (sub-item 2.3) e as tecnologias envolvidas no desenvolvimento do projeto piloto (sub-item 2.4) e os resultados alcançados (sub-item 2.5).

2.1 Modelo de Informação Comum - CIM

O CIM tem suas raízes na iniciativa do EPRI (*Eletric Power Research Institute*) na década de 1980 que buscava o desenvolvimento de uma API (*Application Program Interface*) para centros de controle – CCAPI, visando a troca de dados entre centros de controle com diferentes fornecedores de aplicações EMS (*Energy Management System*).

O CIM é um modelo publicado nas normas IEC 61970-301 e IEC 61968-11 que descreve uma representação de objetos, seus atributos, associações e relações com outros objetos. Os objetos modelados podem ser físicos, como equipamentos de uma rede elétrica, ou podem ser abstratos como objetos utilizados no sistema de informação de consumidores. O principal objetivo do modelo de informação é descrever um domínio sem se restringir à utilização de uma tecnologia específica. No caso do CIM a definição oficial do modelo é feita utilizando a UML (*Unified Modeling Language*) (OMG, 2010).

O modelo está dividido em vários pacotes (Figura 2) , provendo uma visão lógica do domínio de geração, transmissão, distribuição de energia elétrica;

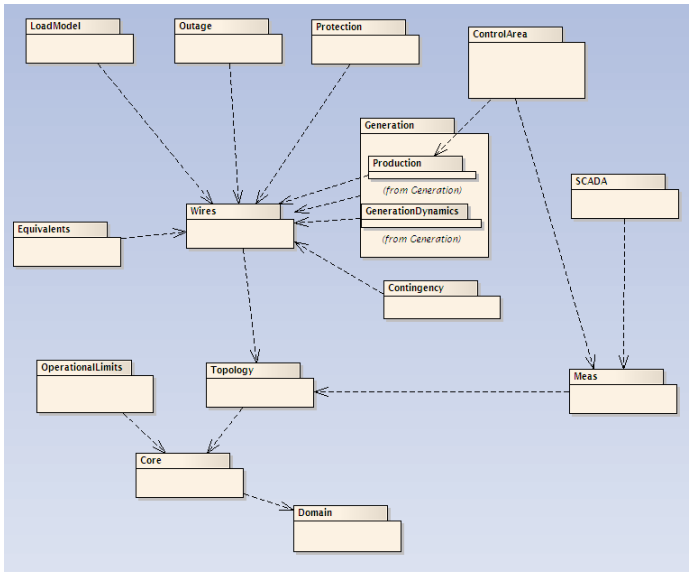


Figura 2 – Pacotes CIM
(IEC 61970 cim_15v01 - IEC 61968 cim_10v30)

O propósito do modelo é ser utilizado na troca entre os agentes de negócio do setor elétrico. Os agentes de negocio não precisam armazenar suas informações nativamente no formato CIM, não

sendo necessárias alterações nas atuais bases de dados¸ no entanto, terão que ter interfaces baseadas em CIM.

2.2 Definição de Interface Genérica - GID

A definição de interface genérica, ou em inglês *Generic Interface Definition* – GID, é um padrão definido nas normas IEC 61970-4xx. Nestas normas são especificadas interfaces de componentes (CIS – *Component Interface Specification*) que definem uma API para acesso aos dados. De forma resumida, são definidas as interfaces que um componente (ou aplicação) deve implementar para trocar informações com outros componentes e/ou para acessar os dados que foram publicados seguindo a forma padrão (IEC 61970-401). Seu objetivo é promover a interoperabilidade de aplicações, tornando-se um mecanismo de grande importância para que haja independência de fornecedores de softwares.

De forma simplificada, as interfaces GID são agrupadas da seguinte forma:

- Interfaces "*Common Services*" – IEC 61970-402: interfaces que fornecem funcionalidades básicas, mas necessárias para as demais partes. São consideradas comuns por englobar um conjunto geral de conceitos que podem ser aplicadas a muitos tipos de sistemas. As especificações destas interfaces (CIS), assim como as demais, se referem a objetos definidos no CIM.
- Interfaces "*Generic Data Access*" (GDA) – IEC 61970-403: fornece um mecanismo de acesso aos dados do tipo "*request/reply*" para as demais aplicações. É importante destacar que esse tipo de interface não é destinada a acessos de dados em tempo real. GDA pode ser classificado como um *Enterprise Information Integrator* (EII).
- Interfaces "*High Speed Data Access*" (HSDA) – IEC 61970-404: especifica uma interface para a transferência eficiente de dados em um ambiente distribuído. Preconiza que pequenas quantidades de dados sejam transferidas com pequeno atraso, mas que também grandes quantidades devam ser transferidas em um tempo curto. A especificação leva em consideração aspectos como a latência causada por uma rede de área local (LAN). Uma aplicação típica seria a publicação e uso de dados de varredura de um sistema SCADA.
- Interfaces "*Generic Eventing and Subscription*" (GES) – IEC 61970-405: especifica uma interface para a transferência eficiente de mensagens de eventos e de reconhecimento de alarme. Da mesma forma como a HSDA, preconiza que pequenas quantidades de dados sejam transferidas com pequeno atraso, mas que também grandes quantidades devam ser transferidas em um tempo curto. Esses requisitos são tipicamente de um sistema SCADA, que atua como um provedor de dados em tempo real a outros sub-sistemas.
- Interfaces "*Time Series Data Access*" (TSDA) – IEC 61670-407: especifica uma interface para a transferência eficiente de dados históricos. Um componente importante nesta abordagem é a utilização de *Data Warehouse*, que atua como um repositório de dados.

No que se refere às normas que especificam os serviços genéricos, ao invés de criar um conjunto completamente novo de APIs, a elaboração do GID se baseou em padrões já consagrados e difundidos na indústria. Cada conjunto de interfaces pode fazer uso de um ou mais padrões oriundos da OMG (*Object Management Group*) ou da OPC Foundation.

2.3 Arquitetura de referência

O projeto piloto em desenvolvimento na Cemig Distribuição S.A baseou-se em uma arquitetura de referência onde os sistemas computacionais da companhia – sistemas legados – pudessem ter suas informações integradas e disponibilizadas através de uma interface padronizada para outros sistemas e usuários.

Uma arquitetura de referência consiste de um conjunto mínimo de conceitos unificados, axiomas e relacionamentos com um domínio de um problema particular, e é independente de padrões específicos, tecnologias, implementações ou outros detalhes concretos. O propósito de um modelo de referência é oferecer um *framework* conceitual comum que possa ser usado consistentemente através e entre diferentes implementações (OASIS, 2006). A arquitetura adotada pelo projeto se baseou na arquitetura de referência SOA (*Service Oriented Architecture*) publicada pela (IBM, 2006a).

2.3.1 Arquitetura orientada a serviços - SOA

A arquitetura orientada a serviços SOA preconiza que as funcionalidades implementadas devem ser disponibilizadas na forma de serviços. O termo serviço se refere ao encapsulamento de módulos de negócio com uma interface que pode ser invocada remotamente. Exemplos de serviços SOA no dominio da Distribuicao: *serviço de transformador* ou *serviço de clientes*.

Nesta abordagem, não interessa como o serviço foi implementado, contanto que ele responda aos comandos da forma correta e em conformidade com os níveis de serviço estabelecidos. Isto significa que o serviço precisa atender os critérios funcionais e não-funcionais estabelecidos nos níveis de serviços, como por exemplo: as metas de DIC (Duração de Interrupção por Unidade Consumidora).

Geralmente iniciativas SOA são originadas nas áreas de negócio, onde a concessionária pode atuar em alterações no processo do negócio como por exemplo melhorias nos serviços de clientes, redução da duração de interrupções, gerenciamento da demanda e etc (EPRI, 2008). Em 2006, uma pesquisa verificou que 63% dos projetos SOA são conduzidos pelas as áreas de negócio - LOB (*Line-of-bussiness*) (IBM 2006b). É importante salientar que SOA não é um produto que possa ser adquirido de um fornecedor e sim um *framework* e orientações para uma arquitetura de sofware (EPRI, 2008).

2.3.2 Barramento de Serviços Corporativo – ESB

Um barramento de serviços corporativos - ESB é comumente utilizado para prover a integração das aplicações e informações. Os serviços são conectados ao barramento que disponibiliza interfaces, ou contratos, acessíveis através de *web services* ou outra forma de comunicação entre aplicações. Essa arquitetura provê um alto grau de reuso, desacoplamento e agilidade, extinguindo as conexões ponto-a-ponto. Somente o ESB não implementa uma arquitetura SOA, mas pode prover a condição para sua implementação (EPRI, 2008).

Entretanto, essa solução arquitetural requer avaliação de atributos de qualidade (Ex.: desempenho, segurança, dentre outros). Devido à latência, excesso de comunicação ou outros problemas advindos da estrutura e composição dos serviços, pode-se ter uma baixa performance. No entanto, o desempenho não é unicamente relacionado à arquitetura do barramento e, portanto, outros fatores precisam ser avaliados, como: definição da granularidade adequada na criação e composição dos serviços; aspectos de arquitetura de segurança, distribuição e gerenciamento (MICROSOFT, 2009).

2.3.3 Descrição da arquitetura adotada

Com base na arquitetura de referência (IBM, 2006a), o projeto piloto construiu a seguinte arquitetura candidata - Figura 3.

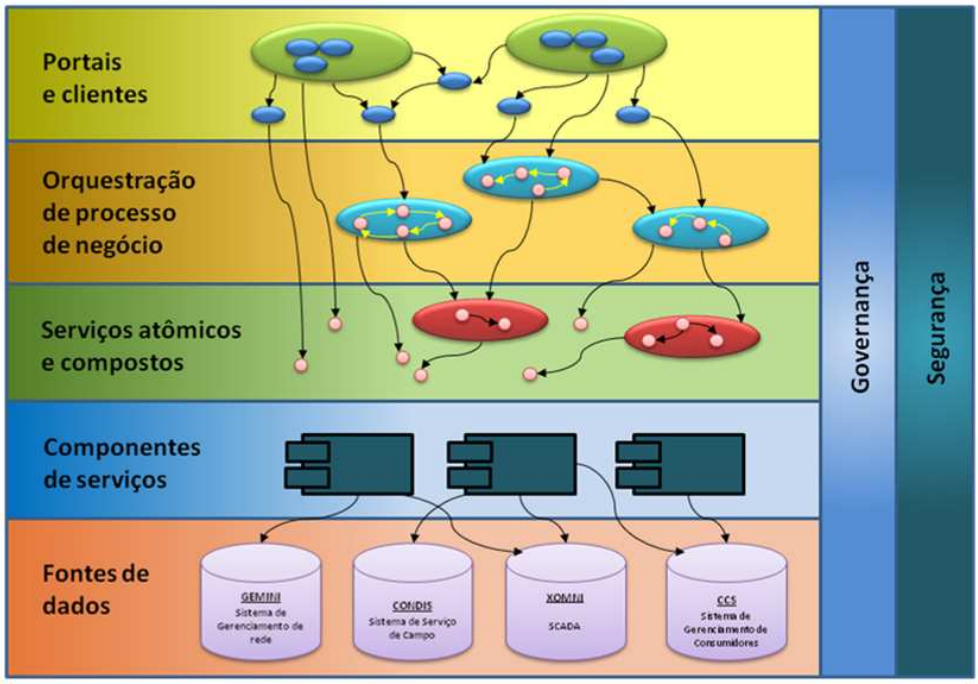


Figura 3 - Arquitetura candidata

A. Fontes de dados

Na camada "Fontes de dados" da Figura 3 estão as bases de dados dos sistemas existentes que suportam as atividades de negócio, como: sistema de gerenciamento de rede, sistema de serviço de campo, SCADA e o sistema de consumidores.

B. Componentes de serviços

A camada de "Componentes de serviços" contém os componentes EJBs (*Enterprise Java Beans*) que provêm a implementação dos serviços. A camada obedece aos contratos de serviço definidos na camada "Serviços Atômicos e Compostos", garantindo o alinhamento entre a implementação de TI com a descrição do serviço. Além disso, asseguram a execução do serviço, com garantia de qualidade e com adesão aos níveis de serviço (*Service-Level Agreements – SLAs*).

Esta camada é constituída por todos os serviços definidos na arquitetura, onde um serviço é uma especificação de funções de TI alinhadas ao negócio. Esta especificação fornece ao cliente detalhes suficientes para consumir as funções de negócio. O conjunto de serviços contidos nesta camada permite utilizar recursos fornecidos por diferentes fornecedores, fazendo SOA ser recomendada para ambientes de TI que possuem hardware e software de múltiplos fabricantes (IBM, 2006b).

Na Figura 3, estão ilustrados serviços atômicos e serviços compostos. Um serviço atômico, conceitualmente, é um componente de software que realiza funcionalidades de negócio ou técnicas, que possui uma complexidade que não se justifica a sua decomposição em componentes menores. Como exemplo: o serviço de cliente. Já um serviço composto é uma estrutura que agrega serviços menores e com menor granularidade, proporcionando outras funcionalidades de negócio ou técnicas. Um serviço composto pode agregar serviços atômicos ou outros serviços compostos. Como exemplo: serviço de triagem de falta. Na Figura 3, os serviços compostos estão delimitados por elipses.

C. Orquestração de processo de negócio

Na camada de "Orquestração de processo de negócio" são abordados os aspectos de gerenciamento de processo de negócios (BPM), que é composto de um ciclo de vida de modelagem, orquestração, implantação e monitoração do processo de negócio. A modelagem de um processo é realizada em uma linguagem que pode ser BPMN (*Bussines Process Modeling Notation*) ou até

mesmo a tradicional UML. Para a orquestração do processo de negócio é utilizado ferramentas com linguagens de programação BPM, como, por exemplo, a BPEL (*Business Process Execution Language*) que permite que os desenvolvedores ou analistas de negócio realizem a orquestração de serviços, trabalhando em um nível de abstração mais alto. A implantação de um processo tem como objetivo disponibilizá-lo para ambiente de produção. Finalmente, a monitoração de um processo permite avaliar indicadores de negocio relacionados a este processo e extrair métricas em tempo real. Como exemplo de processo de negócio citamos a interligação de novo consumidor à rede.

D. Portais e clientes

A camada de "Portais e clientes" permite o acesso aos serviços e a visualização das informações em portais *web*, através de *portlets*. *Portlets* são componentes de software com interface com o usuário que exibem informações. O padrão *Java Portlet Specification* (JSR168, JSR286) permite aos desenvolvedores criarem *portlets* "plugáveis" em qualquer portal que suporte o padrão, proporcionando a interoperabilidade entre diferentes portais *web*, lidando com as questões de customização, apresentação e segurança.

E. Governança e segurança

Na arquitetura SOA, geralmente os custos das primeiras implantações são maiores até o momento em que comece a ocorrer reaproveitamento de serviços. Ao mesmo tempo em que essa arquitetura proporciona uma grande flexibilidade na criação de um novo serviço, gera a necessidade de uma grande atenção na Gerência de Mudança e Garantia de Qualidade de Serviços. O reuso de código está relacionado principalmente à composição de serviços. Nesse cenário, a alteração ou criação de um serviço, caso não seja realizada de forma controlada, pode degradar diversos outros.

Para que essa reutilização seja possível é necessário políticas de Governança, possibilitando controle e gerenciamento dos processos e serviços. Para uma Governança eficaz, devem ser definidas regras para a sua estrutura operacional e organizacional, bem como as normas tecnológicas. Além disso, uma Governança SOA faz o monitoramento da performance dos serviços, gerência o ciclo de vida dos componentes, cuida das políticas de restrição e controle de acesso.

2.4 *Tecnologias utilizadas*

Nesta seção serão explicadas e especificadas as tecnologias utilizadas no projeto, como: ferramentas de desenvolvimento (IDEs), linguagem de programação, servidores de aplicação e entre outros.

Um servidor de aplicação é um programa instalado em um servidor que disponibiliza um ambiente para a instalação e execução de scripts, programas e rotinas para apoio à construção de aplicações, sendo considerado na literatura um software de *middleware*. Inicialmente, o termo servidor de aplicação foi utilizado no contexto de aplicações *web*, mas evoluiu para camadas de serviços muito mais abrangentes. Atualmente, eles implementam serviços de segurança, garantia de disponibilidade (*clustering)*, balanceamento de carga e tratamento de exceções. No mundo de software livre, os servidores Glassfish v2.1 e JBoss AS 4.2.3.GA são os mais conhecidos, sendo o último escolhido para o projeto. Devido a popularização da plataforma Java, o termo servidor de aplicação é frequentemente uma referência a "Servidor de aplicação Java EE" que disponibiliza padrões para os *containers web* e EJB.

O Java EE define o *container web* como um ambiente para execução de aplicações *web* em Java, como s*ervlets*, que são classes Java cujo ciclo de vida é gerenciado pelo *container*. A especificação EJB é uma das várias APIs Java. Seu principal propósito é fornecer uma forma padrão para os códigos necessários em aplicações corporativas, como persistência, controle transacional e concorrência, segurança, dentre outros. Dessa forma, os desenvolvedores ficam livres para se concentrarem no problema do negócio. A especificação define como o componente EJB deve ser

implantado (*deploy*) no *container* EJB do servidor de aplicação. O *container* EJB gerencia o ciclo de vida de componentes EJB.

No projeto, foram desenvolvidos códigos na linguagem Java para acessar a base de dados dos sistemas legados. Estes códigos foram encapsulados em componentes EJBs e publicados no servidor de aplicação. Dessa forma, uma informação de um sistema legado tornou-se disponível para outros sistemas ou usuários que tenham acesso ao servidor de aplicação.

Estes componentes utilizam o recurso de fonte de dados (*data-sources*) que gerenciam o acesso a fonte física dos dados. Este recurso é uma forma padronizada de obter conexões das fontes de dados e ainda implementa *pool* de conexões e transação distribuída. Uma das vantagens de sua utilização, é que as propriedades da fonte de dados podem ser alteradas sem a necessidade de mudança no código, isolando os componentes EJB das fontes de dados.

As fontes de dados dos sistemas legados da concessionária são constituídas principalmente por banco de dados de diversos fornecedores (Ex.: Microsoft SQL Server, Oracle DB, IBM Informix, dentre outros) e também arquivos texto e *sockets*. Para a conexão aos dois últimos, foram utilizados conectores JCA (*Java Conector Adapter*).

Além das interfaces tradicionais (remota e local) para a exposição dos dados através de métodos de um componente EJB é possível a publicação das operações na forma de *web services*, com o uso de anotações Java. Um *web service* é um tipo de API que pode ser acessada por redes, como a Internet, e executar o serviço requisitado no servidor remoto. Sua interface é descrita em um formato definido pelo (W3C, 2010) conhecido como WSDL (*Web Services Description Language*).

Os serviços foram projetados para utilizarem mensagens com parâmetros CIM e também um modelo interno da Cemig. Essa iniciativa se deve ao fato do CIM requerer amadurecimento em algumas áreas principalmente no domínio de distribuição de energia elétrica (GARTNER, 2006). Ao construir duas interfaces para o mesmo serviço, na prática, foi iniciado um movimento de integração de aplicações orientada a serviços, preparando a adoção do CIM em uma escala corporativa sem tantos efeitos colaterais.

A adoção do CIM como modelo padrão da concessionária para a integração de aplicações depende da avaliação dos requisitos de desempenho e de vários outros fatores como: um grande nível de treinamento para desenvolvedores, analistas de negócio, pessoal de operação e executivos, que devem entender como as diversas normas devem ser aplicadas.

A integração sugerida pela (EPRI, 2008) utiliza o envelopamento do CIM através do padrão RDF (*Resource Description Framework*). Entretanto, optou-se pelo uso do CIM em linguagem Java, com o envelopamento em SOAP (HAUSTEIN, 2001), pois este último é um protocolo mais difundido e utilizado em infraestruturas SOA. Para a geração do CIM em linguagem Java os seguintes passos foram realizados:

1. Obtenção do CIM em UML no endereço: http://cimug.ucaiug.org.
2. Criação de plug-in para o EA no Visual Studio.
3. Personalização dos templates de transformações e geração do Enterprise Architecture, pois o modelo que apresentava inconsistências que impediam a geração de código, tais como: atributos com tipos não declarados/inexistentes, classes vazias (sem atributos), utilização de palavras reservadas de linguagem, utilização de caracteres especiais em atributos e enumerações declaradas com membros de diferentes domínios. (Versão analisada: 61970 CIM 15 Versão 01 e 61968CIM 10 Versão 30)

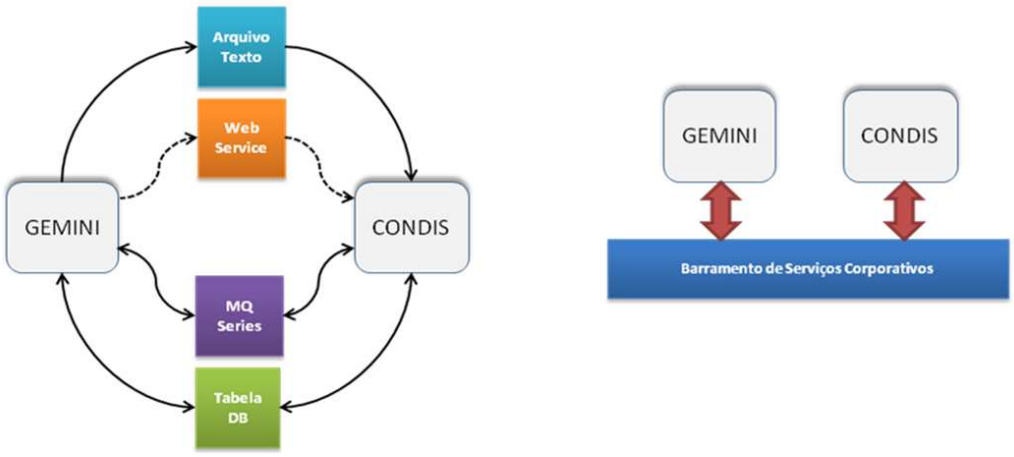
Apesar da importância das interfaces GID, até o momento, o projeto piloto não desenvolveu interfaces neste padrão, visto que o GID está focado na integração de centros de controle (EPRI, 2008), o que não é o caso dos sistemas de distribuição de energia envolvidos nesta etapa do projeto.

Outro fator, é que a definição da arquitetura candidata e os testes tecnológicos puderam ser alcançados sem o uso das interfaces GID.

2.5 *Resultados alcançados*

O processo de triagem de faltas da rede de distribuição de energia elétrica é realizado pelos sistemas de gerenciamento de rede (Gemini) e de serviço de campo (Condis). O Condis recebe as informações da URA (unidade de resposta audível) sobre falta de energia e, através das informações de seqüência elétrica do Gemini, consegue identificar qual o equipamento de proteção possivelmente atuou e encaminha a equipe de campo para solucionar o problema.

Atualmente a conexão entre os dois sistemas é ponto a ponto e parte da base de dados é copiada de um sistema para outro, gerando problemas de duplicação e desatualização de informações. Ambos os sistemas foram desenvolvidos internamente pela Cemig. O sistema de gerenciamento de rede, Gemini, foi implantado em 1998 e escrito em linguagem C++ com banco de dados Microsoft SQL Server. O sistema de serviço de campo, Condis, foi implantado em 1992 e escrito em Java, com banco de dados IBM Informix. Atualmente esses sistemas se comunicam através de troca de mensagens (IBM MQSeries), arquivos texto, compartilhamento de tabela de banco de dados e ainda *Web Services* (em casos de contingência) – Figura 4. Em dias chuvosos, quando ocorre uma maior comunicação entre os sistemas, são aproximadamente 400 requisições por minuto.



Integração atual Integração piloto

Figura 4 – Comparativo entre integração atual e integração piloto

Ao final do projeto, com a construção de adaptadores, os sistemas passarão a publicar serviços superando as dificuldades das diferentes tecnologias de armazenamento de dados e linguagens dos sistemas.

Optou-se por construir serviços com entidades interna da CEMIG visto que as classes semelhantes do CIM ainda não estão maduras o suficiente para transportar todas as informações necessárias para subsidiar esse processo. Essa medida foi adotada para validação da arquitetura e das tecnologias envolvidas, com a visão que para um ambiente de produção interoperável, a adoção de um modelo de informação padronizado e de interfaces genéricas é indispensável. O CIM está sendo estendido pelos grupos de trabalho do comitê TC57 do IEC.

Para validar a escalabilidade da arquitetura proposta e das tecnologias empregadas, foram realizados testes de estresse através de requisições aos serviços. O servidor de aplicação foi instalado em uma estação de trabalho com processador Core 2 Duo E8400@3GHz, 2 GB de RAM, HD de 160GB 7200RPM SATAII. As requisições foram realizadas pela rede local através de uma estação com processador Core 2 Duo E6550@2.33GHz, HD de 250GB 7200RPM SATAII e 3 GB de RAM.

As duas estações estão com o sistema operacional Windows XP Professional com Service Pack 3 e máquina virtual *JRE Kit 6 Update 20*. O serviço consulta um banco de dados Microsoft SQL Server Developer 2005, instalado em um servidor virtual com sistema operacional Windows Server 2003 x64 com processador Dual Core AMD Opteron 2.6GHz e com 1GB de RAM. O banco de dados é uma cópia do banco de produção e está dedicado para este projeto, portanto, não houve concorrência com outras aplicações.

Como diversos clientes podem acessar os serviços simultaneamente, as requisições foram também realizadas através de múltiplas *threads*. As simulações foram organizadas de acordo com sua natureza: simples, onde basicamente ocorre o preenchimento de uma lista de estrutura de dados, como por exemplo, a relação dos consumidores ligados a um determinado ponto físico – Tabela 1; complexas, quando é necessário "caminhar na rede", como, por exemplo, a relação de transformadores que estão à jusante de um determinado equipamento – Tabela 2.

Para garantir que as requisições das diversas *threads* sejam diferentes, foram gerados números aleatórios para preenchimento da lista que realizará a chamada ao serviço. Para cada operação foram feitas 3000 requisições em diferentes combinações (Quantidade de Requisição seqüencial x Quantidade de *threads*) para avaliar o desempenho em diversos cenários. Esse processo foi repetido 20 vezes para ter uma boa confiabilidade estatística dos resultados.

Tabela 1 – Resultado dos testes dos serviços simples

Teste	Threads	Requisições por Thread	Threads	Requisições por Thread	Threads	Requisições por Thread
	10	300	30	100	100	30
	Duração (seg)	Média (Qte/min)	Duração (seg)	Média (Qte/min)	Duração (seg)	Média (Qte/min)
1	194	927,84	30	6.000,00	30	6.000,00
2	175	1.028,57	24	7.500,00	25	7.200,00
3	206	873,79	24	7.500,00	25	7.200,00
4	195	923,08	28	6.428,57	25	7.200,00
5	223	807,17	26	6.923,08	28	6.428,57
6	223	807,17	28	6.428,57	27	6.666,67
7	198	909,09	28	6.428,57	29	6.206,90
8	185	972,97	30	6.000,00	28	6.428,57
9	193	932,64	34	5.294,12	32	5.625,00
10	187	962,57	35	5.142,86	32	5.625,00
11	202	891,09	30	6.000,00	30	6.000,00
12	191	942,41	24	7.500,00	24	7.500,00
13	194	927,84	23	7.826,09	24	7.500,00
14	193	932,64	32	5.625,00	25	7.200,00
15	192	937,50	28	6.428,57	28	6.428,57
16	184	978,26	27	6.666,67	27	6.666,67
17	181	994,48	29	6.206,90	29	6.206,90
18	173	1.040,46	28	6.428,57	29	6.206,90
19	162	1.111,11	31	5.806,45	32	5.625,00
20	135	1.333,33	24	7.500,00	31	5.806,45
Média (Qte/min)		**961,70**		**6.481,70**		**6.486,06**
Desvio Padrão		**112,92**		**773,33**		**630,90**

Nos serviços complexos, não foi possível realizar a simulação com 100 *threads*, devido a problemas de instabilidade de conexão, cujas causas ainda não foram diagnosticadas, mas podem estar relacionadas com o tamanho das estruturas de dados que são necessárias para percorrer na rede. Com a otimização do algoritmo e um servidor corporativo, espera-se que este problema seja solucionado. Assim, até o momento, foi possível simular 50 *threads* para serviços dessa natureza (50 threads com 60 requisições seqüenciais).

Tabela 2 – Resultado dos testes dos serviços complexos

Teste	Threads	Requisições por Thread	Threads	Requisições por Thread	Threads	Requisições por Thread
	10	300	30	100	50	60
	Duração (seg)	Média (Qte/min)	Duração (seg)	Média (Qte/min)	Duração (seg)	Média (Qte/min)
1	244	737,70	161	1118,01	180	1000,00
2	225	800,00	189	952,38	156	1153,85
3	226	796,46	160	1125,00	153	1176,47
4	271	664,21	183	983,61	145	1241,38
5	233	772,53	246	731,71	150	1200,00
6	202	891,09	155	1161,29	179	1005,59
7	207	869,57	200	900,00	143	1258,74
8	235	765,96	231	779,22	158	1139,24
9	248	725,81	168	1071,43	171	1052,63
10	200	900,00	194	927,84	138	1304,35
11	237	759,49	178	1011,24	144	1250,00
12	250	720,00	165	1090,91	153	1176,47
13	248	725,81	144	1250,00	141	1276,60
14	222	810,81	188	957,45	183	983,61
15	231	779,22	190	947,37	137	1313,87
16	200	900,00	148	1216,22	142	1267,61
17	208	865,38	153	1176,47	145	1241,38
18	218	825,69	221	814,48	150	1200,00
19	260	692,31	177	1016,95	188	957,45
20	213	845,07	231	779,22	135	1333,33
Média (Qte/min)		792,36		1.000,54		1.176,63
Desvio Padrão		70,48		151,47		117,57

Tanto nos serviços de natureza simples, quanto nos complexos, com o aumento do número de *threads* e um número menor de requisições seqüenciais, a média de requisições por minuto teve uma melhora significativa, principalmente nos serviços de natureza simples.

3. CONCLUSÕES

Com a necessidade crescente de eficiência operacional e de retorno de valor aos acionistas, as concessionárias estão buscando melhorias nos seus processos de negócio. Como as melhorias incrementais possibilitadas pelos sistemas compartimentalizados já foram exauridas (GARTNER, 2006), as integrações baseadas em modelos, como o CIM, e com arquitetura orientada a serviços, é que possibilitarão integrações mais flexíveis, escaláveis e ágeis para as mudanças nos processos do negócio (GARTNER, 2006).

O projeto piloto, ao realizar uma integração de sistemas baseadas em modelos de informação, demonstra a viabilidade da arquitetura candidata. Os resultados obtidos neste projeto demonstraram a escalabilidade da arquitetura adotada e das tecnologias empregadas. O requisito de desempenho foi atendido com boa margem de segurança, mesmo utilizando máquinas de trabalho comuns. Apesar de todos os softwares utilizados serem gratuitos e de código aberto (*open-source*), o desenho do projeto permite a adoção de ferramentas proprietárias (Ex: IBM, Oracle, Tibco), pois a maior parte da tecnologia envolvida possui padrões independentes de fornecedor.

Uma vantagem do uso de softwares proprietários está no grau de produtividade que essas ferramentas oferecem. Ainda, para o estabelecimento de uma arquitetura executável (ambiente de produção) é necessário a realização de mais testes para os requisitos não funcionais (exemplo: escalabilidade e disponibilidade) e a formalização do processo de integração envolvendo casos de uso, diagramas de seqüência, projeto e construção das mensagens e interfaces, além da governança dos artefatos (EPRI, 2008).

O objetivo último da integração total entre os sistemas de uma concessionária está distante, pois diversos fatores são necessários para a adoção do CIM como padrão de gerenciamento das informações da empresa (EIM), como por exemplo: amadurecimento tecnológico para suportar a integração de todos os sistemas, evolução dos padrões CIM em algumas áreas, treinamento nos novos padrões para os desenvolvedores, analistas de negócios e pessoal de operações, contratação de consultores TI, alinhamento estratégico e comprovação do retorno dos investimentos.

Uma adoção antecipada do CIM, no entanto, poderá reduzir significativamente o risco dos custos adicionais em implementações tardias (GARTNER, 2006). Com o desenvolvimento das redes inteligentes (*Smart Grids*), principalmente as aplicações voltadas para o consumidor, como controle de cargas e de demanda, automação predial, uso de veículos elétricos e geração distribuída de energia. (GARTNER, 2009).

4. REFERÊNCIAS BIBLIOGRÁFICAS

1. EPRI. Enterprise Service Bus Implementation Profile - Integration Using IEC 61968, Maio 2009.
2. EPRI. The Common Information Model for Distribution - An Introduction to the CIM for Integrating Distribution Applications and Systems, Novembro 2008.
3. EPRI. The Integrated Energy and Communication System Architecture – Volume II: Funcional Requirements, 2003.
4. GARTNER. CIM Adoption Is on the Rise, So Energy and Utility IT Leaders Should Prepare, 14 de Julho 2006.
5. GARTNER. CIM User Survey: Model-Driven Integration Standards Are Used and Useful in Electric Utilities, 12 de Fevereiro 2009.
6. GARTNER. Smart Grid Intensifies CIM Activity, 22 de Dezembro 2009.
7. HAUSTEIN, Stefan. Semantic Web Languages: RDF vs. SOAP Serialisation - 2001. Acesso em 25/02/2010, disponível em: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.23.2537
8. IBM. Arquitetura Orientada a Serviços – SOA - Infraestrutura para a Inovação - 2006b. Acesso em 19/02/2010, disponível em: http://www.pr.senai.br/posgraduacao/uploadAddress/Introducao%20ao%20SOA%5B31574%5D.pdf
9. IBM. Design an SOA solution using a reference architecture. 2006a. Acesso em 25/02/2010, disponível em: http://www.ibm.com/developerworks/library/ar-archtemp/
10. LIMA, Luiz C. e SANTOS, Hiram T. Aplicação do padrão CIM em centros de controle e para integração corporativa – 2007.
11. MCMORRAN, Dr.Alan W. - An Introduction to IEC 61970-301&61968-11 – The Common Information Model – University of Strahclyde – January 2007.
12. MICROSOFT. Conhecendo melhor as Capacidades do Enterprise Service Bus – Junho 2009. Acesso em 22/02/2010, disponível em: http://msdn.microsoft.com/pt-br/library/dd920288.aspx
13. OASIS. Modelo de Referência para Arquitetura Orientada a Serviço 1.0. 2006. Acesso em 19/02/2010, disponível em: http://www.pcs.usp.br/~pcs5002/oasis/soa-rm-csbr.pdf
14. OMG. Object Management Group - Unified Modeling Language. Acesso em 25/02/2010, disponível em: http://www.uml.org/
15. W3C. World Wide Web Consortium – 2010. Acesso em 25/02/2010, disponível em: http://www.w3c.org/