FRAMEWORK DE AUTOMAÇÃO DA ARCELORMITTAL TUBARÃO - PROJETO PILOTO[1]

Weslley Canal Matede[2]
Daniel Carvalho Pedrini[2]

Resumo
Dentro do contexto do Plano Diretor de Tecnologia de Automação da ArcelorMittal Tubarão, identificou-se a necessidade de criação de um framework que suportasse o desenvolvimento de novos sistemas de nível 2 de Automação. Este framework foi concebido para permitir uma maior independência dos sistemas de Automação em relação ao hardware nos quais são executados, bem como permitiu a implementação de funcionalidades para agilizar o processo de desenvolvimento de novos sistemas. As funcionalidades fornecidas mantêm o foco do desenvolvimento concentrado no projeto do software e suas regras de negócio, tornando o processo de desenvolvimento mais eficiente e padronizado. Este trabalho apresenta a experiência adquirida no primeiro projeto, sistema de automação para a Estação de Dessulfuração em carro torpedo, desenvolvido sobre o framework. São apresentadas também as funcionalidades presentes no framework que contribuíram para desenvolvimento do sistema.
Palavras-chave: Automação; Framework de automação; Java; OPC

AUTOMATION FRAMEWORK OF ARCELORMITTAL TUBARÃO – PROJECT PILOT

Abstract
In the context of the Master Plan of Automation Technology of ArcelorMittal Tubarão, it was identified the need to create a framework that supports the development of new level 2 automation systems. This framework was designed to allow greater independence of Automation systems regarding the hardware on which they run, as well as allowed the implementation of functionalities to streamline the process of development of new systems. The functionality provided remains the focus of development concentrated on software design and their business rules, making the development process more efficient and standardized. This paper presents the experience gained in the first project, automation system for Desulphurization Plant, developed on the framework. It is also presented the features present in the framework that contributed for the development of the system.
Keywords: Automation; Automation framework; Java; OPC.

[1] *Contribuição técnica ao 17° Seminário de Automação e TI Industrial, 24 a 27 de setembro de 2013, Vitória, ES, Brasil.*

[2] *Cientista da Computação – Especialista em Automação (ArcelorMittal Tubarão)*

1 INTRODUÇÃO

O parque tecnológico de Automação da ArcelorMittal Tubarão, compostos por Servidores Alpha, em fim de vida útil, com sobressalentes escassos, bem como o avanço tecnológico que nos fornece hardwares mais robustos e soluções de armazenamentos mais rápidas e confiáveis, além de ferramentas e linguagens de desenvolvimento mais avançadas, mostraram à empresa a necessidade de uma atualização tecnológica para garantir a longevidade dos seus sistemas de automação.
Com base nesses fatos, foi realizado um estudo para definir uma arquitetura adequada para essa atualização tecnológica. A partir do estudo, optou-se pela linguagem Java para implementação de um framework de automação com uma arquitetura simples e flexível, com plataforma e programas expansíveis e extensíveis e com robustez e desempenho, além de dar suporte a componentes e subsistemas para atendimento de requisitos das várias áreas operacionais da empresa.
O objetivo desse trabalho é mostrar a arquitetura na qual o Framework de Automação foi desenvolvido e como contribuiu no desenvolvimento do novo Sistema da Dessulfuração em Carro Torpedos.

2 O PROCESSO DE DESSULFURAÇÃO EM CARRO TORPEDOS

O processo de dessulfuração em carro torpedos na ArcelorMittal Tubarão é feito em duas linhas de produção distintas que podem ou não funcionar simultaneamente. Após sair do Alto-forno, o carro torpedo é pesado e encaminhado à planta de dessulfuração. O cálculo da quantidade de mistura dessulfurante necessária é realizado a partir do teor de enxofre do gusa, analisado via LECCO (equipamento de análise química), e do teor de enxofre objetivado após a dessulfuração. O *set-point* de quantidade de material dessulfurante é enviado ao Nível 1 (PLC), que dá início efetivamente ao processo, que consiste em imergir uma lança dentro do carro torpedo para injetar a mistura dessulfurante.
Após o término da injeção uma nova análise é realizada. Se o enxofre visado for atingido, faz-se a liberação do carro torpedo. Caso contrário, um novo cálculo é realizado e uma nova injeção é executada. Apesar de ser raro, esse processo pode-se repetir até que o teor de enxofre atinja a meta desejada.

3 O SISTEMA ANTIGO

O sistema antigo foi projetado em duas configurações, desenvolvimento e produção, e era executado no seguinte ambiente:
Hardware:

- HP Alpha Server DS-20E 500MHz

Software Básico:

- OpenVMS AXP versão 7.3-2
- Oracle RDB versão 7.0-63
- BASESTAR Classic para OpenVMS AXP versão 3.4
- BASESTAR DAS para AllenBradley Interchange versão 3.4
- SL-GMS versão 5.3
- DEC Message Queue versão 5.0-20
- DEC C/C++ Compiler para OpenVMS AXP versão 5.5-002
- TCP/IP versão 5.4

O sistema antigo possuía a seguinte arquitetura:

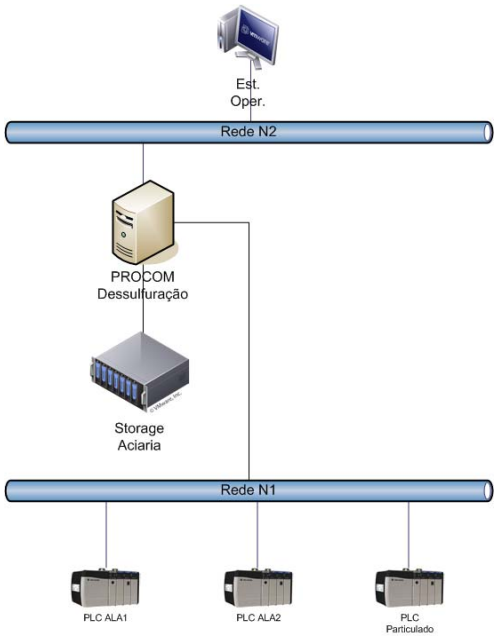

Figura 1 – Arquitetura do Sistema Antigo

4 O FRAMEWORK DE AUTOMAÇÃO

Segundo Mattsson,[1,2] um framework é uma arquitetura desenvolvida com o objetivo de atingir a máxima reutilização, representada como um conjunto de classes abstratas e concretas, com grande potencial de especialização.
Framework se diferencia de uma simples biblioteca, pois esta se concentra apenas em oferecer implementação de funcionalidades, sem definir a reutilização de uma solução de arquitetura.
O conceito do Framework de Automação foi agrupar todas as características em comum dos atuais sistemas de Nível 2, e disponibilizá-las em forma de um framework. Dessa forma, ao desenvolver uma aplicação utilizando esse Framework, o foco fica voltado necessariamente às regras de negócio, pois ele oferece muito mais que apenas uma simples biblioteca, ele oferece também uma arquitetura para desenvolver de forma rápida e eficiente sistemas, através da padronização, da reutilização de código-fonte e concentrando-se mais com a abstração de soluções do problema a ser tratado pelo sistema.
Ele foi projetado para ser usado de forma simples, sem conhecimento algum da tecnologia que está sendo empregada por trás de cada método executado, apoiando assim o desenvolvimento e implantação de novos sistemas de Nível 2 e também na migração dos sistemas hoje existentes.
Ele disponibiliza funcionalidades que cobrem as necessidades de negócio, mas com a habilidade única de alavancar, substituir ou estender vários aspectos da solução, incluindo interfaces, subsistemas e modelos de dados.
A solução combina seu foco na arquitetura com funcionalidades avançadas de forma a acelerar a entrada em operação de soluções escaláveis, enquanto permite a integração e desenvolvimento de sistemas flexíveis de modo a suprir a necessidade dos diversos sistemas existentes e de novas implementações.
O Framework foi modelado utilizando arquitetura MVC (Model-View-Controller). Fowler[3] descreve que: A abordagem multicamadas com o padrão MVC proporciona

a separação entre a lógica da aplicação e a interface com o usuário, considerada uma boa prática de design de software.
Dessa forma, o Framework fornece total separação entre a camada de negócio e a camada de visualização, o que torna o software mais manutenível, aumenta a visibilidade da camada de negócio e mantém clara a separação entre persistência e interface.
Para fortalecer ainda mais a arquitetura MVC, o framework implementa também o *Design Pattern DAO (Data Access Object)* que permite separar regras de negócio das regras de acesso a banco de dados. Por utilizar a arquitetura MVC, todas as funcionalidades de bancos de dados, tais como obter as conexões, mapear objetos Java para tipos de dados SQL ou executar comandos SQL, são feitas por classes de DAO. Dessa forma, consegue-se abstrair todo o contexto relativo ao banco de dados à lógica de negócio da aplicação, permitindo que mudanças no banco de dados não afetem as regras de negócio.
Por se tratar de um Framework, ele fornece *classes* que, quando herdadas, disponibilizam uma variedade de funcionalidades que permitem o desenvolvimento de novos processos para atender a qualquer área do processo produtivo.
Outro ponto importante a ressaltar é que toda comunicação entre os processos que compõem sua estrutura é feita através de RMI (Remote Method Invocation).

> “Uma das principais vantagens do RMI é sua capacidade de baixar o código de um objeto, caso a classe desse objeto não seja definida na máquina virtual do receptor. Os tipos e o comportamento de um objeto, previamente disponíveis apenas em uma máquina virtual, agora podem ser transmitidos para outra máquina virtual, possivelmente remota. Essa funcionalidade do RMI permite que o código da aplicação seja atualizado dinamicamente, sem a necessidade de recompilar o código.”(4)

Mesmo possuindo toda comunicação entre processos baseada em RMI, é abstraído do desenvolvedor o conhecimento de RMI, pois a comunicação entre os processos é feita através de um método remoto disponibilizado pela interface de cada processo da aplicação. Fica a cargo do desenvolvedor apenas definir o nome do evento, os dados que serão transmitidos e o processo destino com o qual ele deseja se comunicar.
Para atender ao processo de automação, onde constantemente se atua em variáveis do controle do processo, foi implementado no Framework uma camada de acesso aos dados de Nível 1. Dessa forma, decidiu-se criar essa estrutura de dados em memória (BDTR – Banco de Dados em Tempo Real) para manter o desacoplamento entre a aplicação e a forma de acesso aos dados de Nível 1 (OPC, Serial, etc...). Assim, quando da necessidade de escrita ou leitura de dados de Nível 1, todo o acesso é feito através da BDTR, deixando a aplicação independente de um conector específico. Atualmente o Framework oferece o conector OPC para acesso aos dados de Nível 1. A Figura 2 representa a arquitetura adotada para a comunicação com o Nível 1.

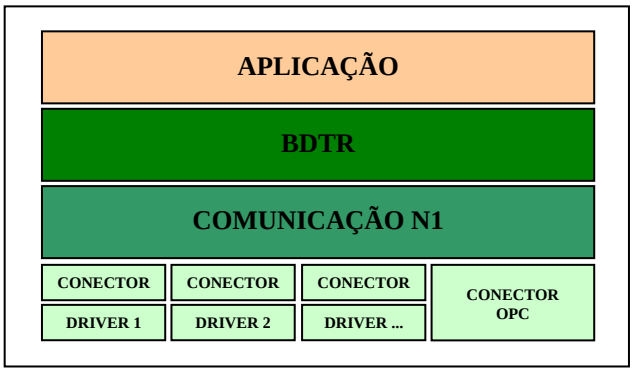

Figura 2 – Comunicação Nível 1.

Além de todas as características já mencionadas acima, resumidamente, o Framework de Automação disponibiliza em seu núcleo os seguintes recursos:

- Gestão de Usuários
- Gestão de Paradas
- Gestão de Alarmes e Eventos
- Gestão de Turnos
- Gestão de Tempos
- Gestão de Log
- Gestão da Configuração
- Gestão de Processos
- Gestão da Comunicação (Nível 1, Nível 2, Nível 3 e entre processos)
- Gestão de Acesso a Dados (Design Pattern DAO + Hibernate)
- Gestão de Interface com o Usuário

Os Sistemas desenvolvidos ou migrados para a nova arquitetura utilizando o Framework de Automação terão disponíveis todos os recursos listados acima.
No próximo capítulo será apresentada, de uma forma mais detalhada, a arquitetura do Novo Sistema da Dessulfuração em Carros Torpedos.

5 ARQUITETURA

Um dos pontos fortes da migração para a nova arquitetura de software baseada no Framework de Automação é a possibilidade de buscar novas soluções de hardware, que possibilitem aumentar o desempenho, a segurança e a disponibilidade dos sistemas.
A seguir são detalhadas as arquiteturas de software e de hardware do novo Sistema da Dessulfuração em Carros Torpedos baseada no Framework de Automação.

5.1 Software

O Modelo abaixo representa de uma forma geral como foi estruturado o novo Sistema da Dessulfuração em Carro Torpedos baseado no Framework de Automação.
Basicamente o Sistema da Dessulfuração em Carro Torpedos possui 7 (sete) processos que são executados no servidor:

- ComunicacaoTerminais – Responsável por receber as requisições dos clientes e direcioná-las as suas respectivas classes de negócio.

- ComunicacaoN1 – Responsável por tratar todos os eventos que são disparados pelo Nível 1 da Dessulfuração.
- ProcessoHistoriador – Responsável por enviar para a base de dados históricos todas as tags configuradas. O valor da tag pode ser o valor instantâneo ou pode ser a média de um conjunto de valores.
- ProcessoComunicacaoOPC – Responsável por ler e escrever dados no servidor OPC via BDTR e também por escrever dados na BDTR.
- ProcessoGestaoParadas – Responsável por abrir e fechar no sistema as paradas de produção, sejam elas manuais ou automáticas.
- ComunicacaoBMQ – Responsável por receber e enviar as mensagens via *Bea MessageQ* .
- ProcessoMonitor - Responsável por partir e manter os demais processos em funcionamento. Ele identifica os processos que apresentam problemas e os reinicia.

A aplicação cliente possui um único processo que é executado na estação de operação. A partir da aplicação cliente, o usuário tem acesso às telas do sistema. O processo cliente é responsável por enviar os eventos dos terminais de operação para o servidor e quem recebe os eventos contendo as respostas aos pedidos do cliente.

A Figura 3 representa graficamente o modelo de classes macro do Sistema da Dessulfuração em Carros Torpedos que utilizou como base o Framework de Automação.

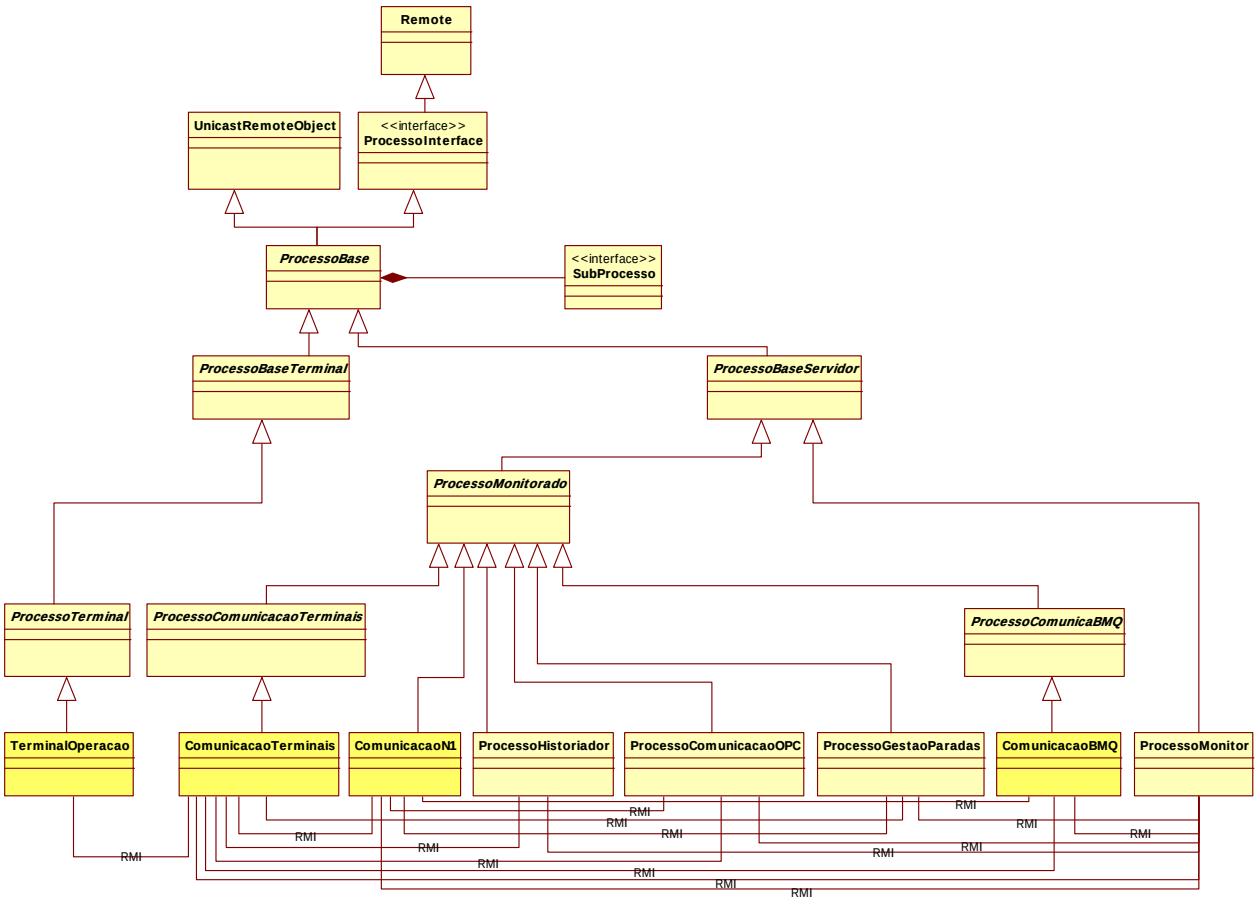

Figura 3 – Diagrama de Classes.

5.2 Hardware

A arquitetura atual é composta por:

- Servidor de Produção: Servidor de aplicação e banco de dados principal;
- Servidor de Contingência: Servidor de aplicação e banco de dados de contingência e desenvolvimento;
- Servidores OPC: Dois servidores OPC trabalhando em redundância.

Essa arquitetura tem como vantagem não compartilhar recursos dos servidores de aplicação e banco de dados com outros sistemas, pois cada sistema possui seu próprio servidor. Isso facilita a manutenção do sistema, pois precisaria somente da parada da planta da Dessulfuração para realizar essa manutenção.
Como desvantagem, os custos de implantação e gerenciamento da solução são maiores, devido ao maior número de ativos a serem adquiridos e gerenciados.

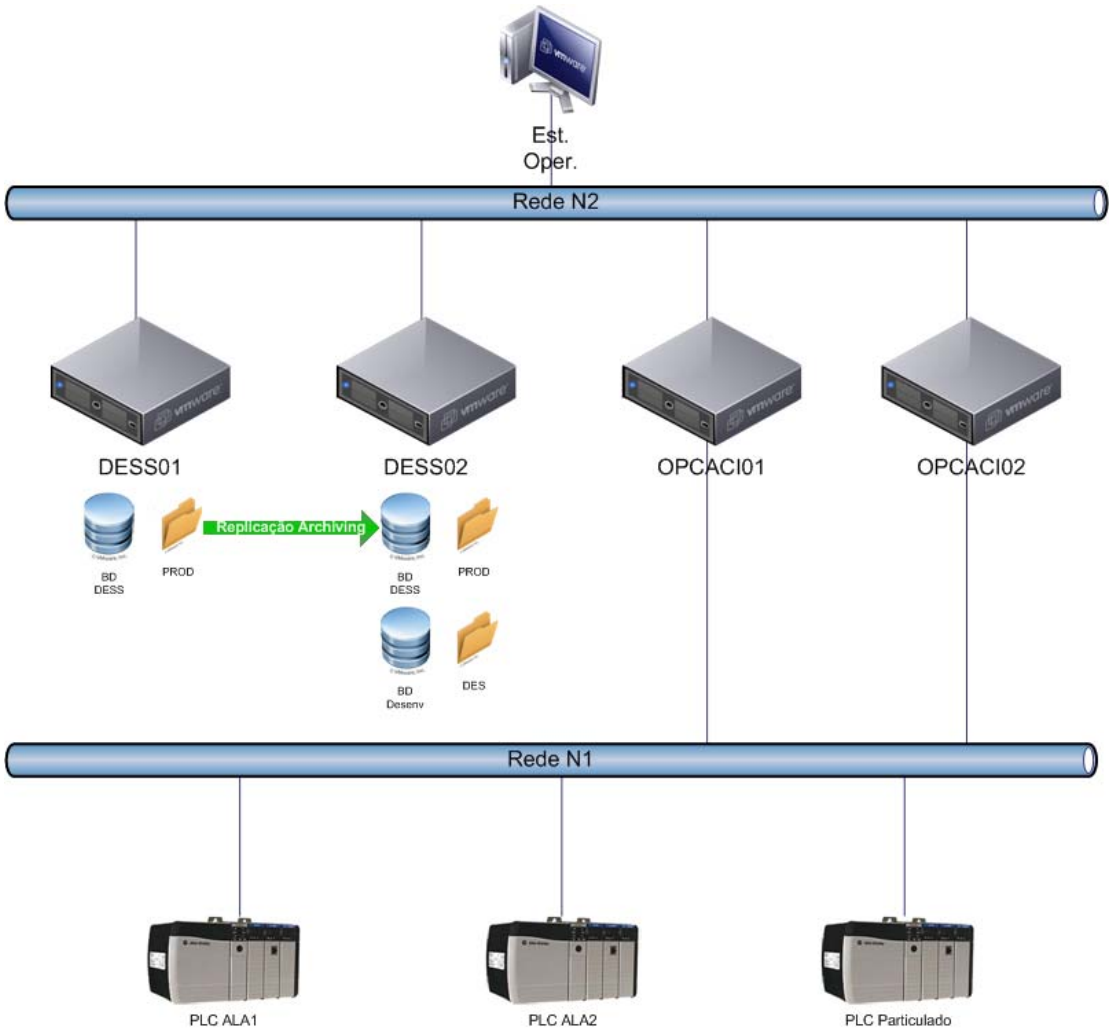

Figura 4 – Arquitetura atual.

A arquitetura a ser implementada futuramente é constituída de:

- Servidor de Produção Virtual: Servidor de Aplicação da produção;
- Servidor de Desenvolvimento Virtual: Servidor de Aplicação de desenvolvimento;

- Servidor de Banco de Dados: Estrutura de banco de dados Centralizado, com mecanismos de alta disponibilidade baseado no *Data Guard* da Oracle;
- Servidores OPC: Dois servidores OPC trabalhando em redundância;

Essa arquitetura tem como vantagem os mecanismos de alta disponibilidade já implementados no ambiente, como:

- *High Availability VMware*: Mecanismo que garante alta disponibilidade para o servidor de produção virtual. Esse mecanismo garante a alta disponibilidade em caso de falha do servidor físico de forma automática, reiniciando o servidor virtual em outro servidor físico;
- *vSphere Replication*: Mecanismo que garante a replicação da máquina virtual de produção para outro *storage*, garantindo o retorno do servidor de produção em caso de falha no *storage*;
- *Oracle Data Guard*: Mecanismo de alta disponibilidade do banco de dados, para caso de falha no servidor e *storage*, onde o mesmo realiza o sincronismo dos dados periodicamente para outro site.

Como desvantagem, essa arquitetura é compartilhada com outros sistemas. Com isso, caso seja necessário realizar manutenção que necessite da parada do ambiente, o mesmo somente poderá ser realizado em parada conjunta com outras plantas, para não afetar a disponibilidade do ambiente para a produção.

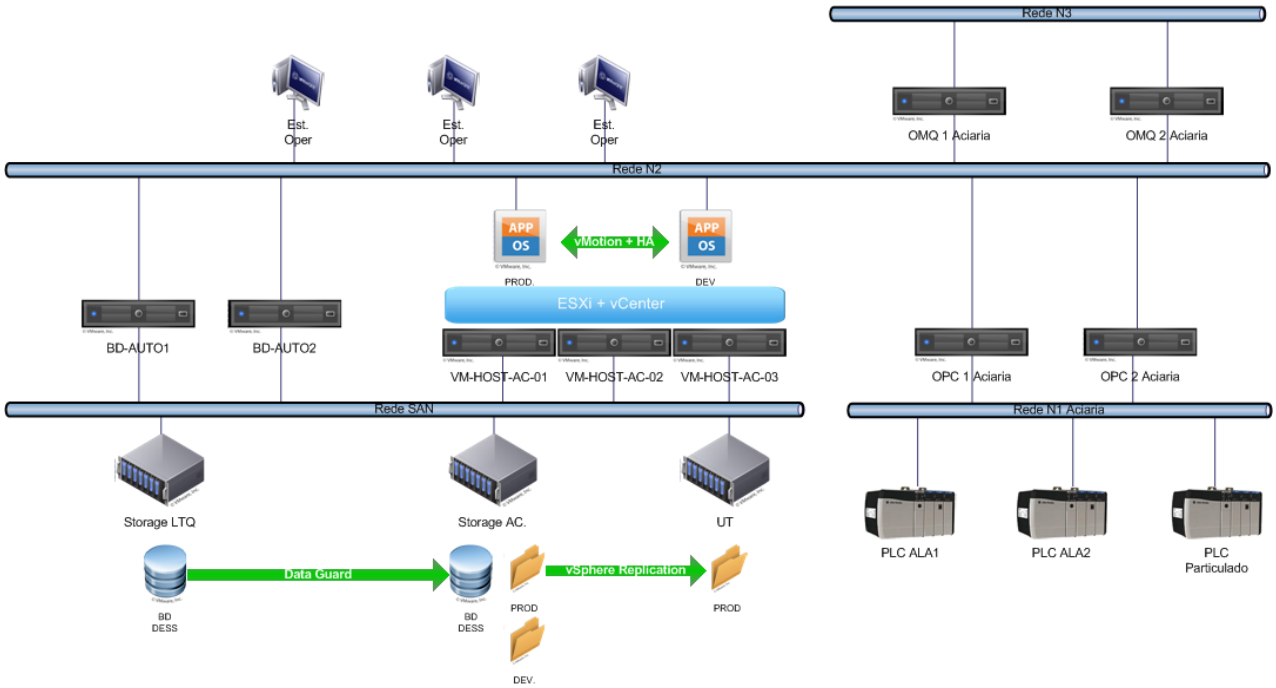

Figura 5 – Arquitetura proposta para o futuro

6 RESULTADOS

Como o Framework de Automação já fornece diversas funcionalidades que o sistema antigo da dessulfuração possuía, durante a fase de projeto do software gastou-se mais tempo com as regras de negócio e melhorias sugeridas pela área operacional.

Outro fato importante foi a facilidade com o qual o Framework possibilitou a configuração da comunicação com o Nível 1 e os demais Sistemas de Nível 2 e Nível 3.
Conforme apresentado no item anterior, uma evolução da arquitetura do sistema será a utilização da virtualização que produzirá os benefícios citados.
Por se tratar de um Projeto Piloto, a dúvida que se tinha era se o sistema se manteria estável após a migração para a nova arquitetura de software e hardware. Decorridos 12 meses após implantação, verificou-se um excelente desempenho apresentado pelo sistema, não apresentando nenhuma ocorrência grave nesse período.

7 CONCLUSÃO

A partir do término desse primeiro projeto, iniciaram-se frentes visando generalizar e padronizar ainda mais o Framework. A idéia é que cada área (Redução, Aciaria e LTQ) identifique dentro do seu domínio de negócio classes “genéricas”, de forma a montar uma ontologia, por exemplo, Ontologia da Aciaria, contendo suas classes, comportamentos e relacionamentos segundo seu domínio de negócio, pois para Grubber:[5] [...] o principal propósito da construção de ontologias é permitir compartilhamento e reutilização de conhecimento.
A Implantação do Projeto da Dessulfuração em Carros Torpedos utilizando o Framework de Automação foi um marco inicial para a consolidação da nova metodologia de desenvolvimento de software, proporcionando uma grande troca de conhecimento e um aprimoramento técnico em relação às ferramentas utilizadas no projeto.

REFERÊNCIAS

1 MATTSSON, M. Object-oriented Frameworks - A survey of methodological issues, Licentiate Thesis, Department of Computer Science, Lund University, CODEN: LUTEDX/(TECS-3066)/1-130/(1996), also as Technical Report, LU-CS-TR: 96-167, Department of Computer Science, Lund University, 1996.
2 MATTSSON, M. Evolution and Composition Object-Oriented Frameworks, PhD Thesis, University of Karlskrona/Ronneby, Department of Software Engineering and Computer Science, 2000.
3 FOWLER, M. Patterns of Enterprise Application Architecture. Addison Wesley, 2002.
4 An Overview of RMI Applications. Disponível em: http://docs.oracle.com/javase/tutorial/rmi/overview.html. Acesso em: 21 mai. 2013.
5 GRUBER, T. Ontology. http://tomgruber.org/writing/ontology-definition-2007.htm. Acesso em: 21 mai. 2013.