FACULDADE DE ADMINISTRAÇÃO E NEGÓCIOS DE SERGIPE – FANESE
CURSO DE ENGENHARIA DE PRODUÇÃO

MARCIO KASTER DA SILVA

GESTÃO DE CONFIGURAÇÃO DE APLICATIVOS DE AUTOMAÇÃO - em uma empresa de petróleo do nordeste Brasileiro

Aracaju – SE
2013.2

MARCIO KASTER DA SILVA

GESTÃO DE CONFIGURAÇÃO DE APLICATIVOS DE AUTOMAÇÃO - em uma empresa de petróleo do nordeste Brasileiro

Monografia apresentada à Banca examinadora da Faculdade de Administração e Negócio de Sergipe - FANESE, como requisito parcial e elemento obrigatório para obtenção do grau de bacharel em Engenharia de Produção.
Orientador: Esp. Josevaldo dos Santos Feitoza.

Coordenador: Msc. Alcides A. Araújo Filho

Aracaju – SE
2013.2

FICHA CATALOGRÁFICA

S586a SILVA, Marcio Kaster da

Gestão de Configuração de Aplicativos de Automação – em uma empresa de petróleo do nordeste brasileiro/Marcio Kaster da Silva. Aracaju, 2013.68 f.

Monografia (Graduação) – Faculdade de Administração e Negócios de Sergipe. Departamento de Engenharia de Produção, 2013.

Orientador: Prof. Esp. Josevaldo dos Santos Feitoza

1. Subversion 2. TortoiseSVN 3. Controle de Versão
4. Automação I. TÍTULO.

CDU 658.5: 681.5 (813.7)

MARCIO KASTER DA SILVA

GESTÃO DE CONFIGURAÇÃO DE APLICATIVOS DE AUTOMAÇÃO - em uma empresa de petróleo do nordeste Brasileiro

Monografia apresentada à Banca examinadora da Faculdade de Administração e Negócio de Sergipe - FANESE, como requisito parcial e elemento obrigatório para obtenção do grau de bacharel em Engenharia de Produção no período de 2013.2.

_____

Prof. Esp. Josevaldo dos Santos Feitoza.
Orientador

_____

Prof.
1º Examinador

_____

Prof.
2º Examinador

Aprovado (a) com média:_____

Aracaju (SE), ___ de _____ de 2013.

Dedico este trabalho a minha amada esposa!

RESUMO

O presente trabalho de conclusão de curso tem como objetivo implementar ferramenta de controle automático dos itens de configuração, no processo de *Gestão de Configuração dos Aplicativos de Automação*, numa empresa de petróleo do nordeste Brasileiro. As ferramentas utilizadas para controle de versão foram o Subversion e o Tortoise SVN. Foi criado um sistema piloto para avaliar a viabilidade do uso da ferramenta no ambiente de automação da empresa e desenvolvido um plano de ação para implantar as mesmas. O plano de ação viabilizou a implantação da ferramenta de controle de versão com sucesso. O subversion possibilitou nos ultimo cinco meses o controle de 761 revisões totalizando 65117 arquivos modificados. Demonstrando um ganho significativo em controle para a empresa, visto que o controle atual via planilha é ineficiente.

Palavras-chave: Subversion. TortoiseSVN. Controle de versão. Automação.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos a tecnologia avançou significativamente e, em paralelo, os projetos de software de automação aumentaram constantemente de tamanho e complexidade. Proporcionalmente a este avanço, o número de problemas enfrentados no desenvolvimento do software de Automação também aumentou.

Durante o ciclo de vida de um projeto, o software Aplicativo pode passar por vários processos de desenvolvimento realizado por diferentes profissionais. O software é altamente propenso a mudanças devido a sua natureza intangível, mutável e complexa. Sendo assim, as mudanças são inevitáveis e constituem um grande fator de dificuldade em projetos de software de automação. Devido às mudanças se torna necessário um controle eficiente sobre as versões.

A empresa analisada possui uma complexa rede de automação que exige um sistema eficaz de controle de versão, mas a mesma tem um controle via planilha. O objetivo desta pesquisa é implementar uma ferramenta de controle centralizada que possibilite o versionamento automático a cada modificação implementada.

1.1 Caracterização da Empresa

A Petrobras é líder no setor petrolífero brasileiro e expande as operações para estar entre as cinco maiores empresas integradas de energia até 2020.

Marcada por uma trajetória de superação de desafios que conduziu a companhia a avanços tecnológicos significativos, como a conquista da liderança em exploração e produção de petróleo em águas profundas e a descoberta de óleo e gás na camada pré-sal, a Petrobras é hoje a maior companhia da América Latina.

A empresa integrada de energia atua, de forma rentável com responsabilidade social e ambiental buscando a ecoeficiência nos processos e produtos. Está presente em 27 países, além de manter atividades na maior parte dos estados do Brasil, e tem ações negociadas nas principais bolsas de valores do mundo.

É sociedade anônima de capital aberto que tem como acionista majoritário o governo do Brasil. A Petrobras atua nos seguintes setores: exploração

e produção, refino, comercialização e transporte de óleo e gás natural, petroquímica, distribuição de derivados, energia elétrica, biocombustíveis e outras fontes renováveis de energia.

Criada em 3 de outubro de 1953 pelo então presidente Getúlio Vargas, a companhia vem, desde então, contribuindo sistematicamente para o desenvolvimento do país. Do salto tecnológico que representou a exploração em águas profundas à descoberta das imensas reservas de óleo e gás na camada pré-sal, passando pela conquista da autossuficiência, a Petrobras impulsiona o crescimento do país por seis décadas.

A descoberta de uma nova província petrolífera na Bacia de Sergipe-Alagoas é o capítulo mais recente da história de sucesso iniciada pela Companhia nos dois Estados no fim da década de 50.

A Unidade enfrenta agora novos desafios em mais um capítulo dessa trajetória de sucessos: a descoberta, em outubro de 2010, de uma província petrolífera em águas ultraprofundas da costa sergipana.

Para chegar ao estágio atual, a Unidade venceu inúmeros desafios. Desde a descoberta do petróleo em terras sergipanas, no ano 1959, a Petrobras tem contribuído fortemente com a economia dos estados de Sergipe e Alagoas. O Campo de Carmópolis, descoberto em 1963, é o maior campo terrestre do país, em volume recuperável de óleo petróleo. Cinco anos depois outra excelente notícia: descobre-se em águas sergipanas o primeiro campo de petróleo na plataforma continental brasileira, batizado de Guaricema, campo que possibilitou a habilitação dos primeiros técnicos brasileiros na prospecção de petróleo no mar, o que faz hoje, da Petrobras uma das mais importantes empresas mundiais no setor.

De lá pra cá foram muitas as conquistas da Petrobras em Sergipe e Alagoas, ressaltadas pela vocação ao pioneirismo da Unidade na exploração e produção de petróleo no Brasil.

A Engenharia de Instalações, Processamento e Automação (E&P-NNE/UO-SEAL/ENGP/EIPA) presta consultoria técnica e elaboração de projetos para instalações de sistemas de separação, processamento, tratamento, injeção, descarte, armazenamento e transferência de fluidos da Unidade Operacional.

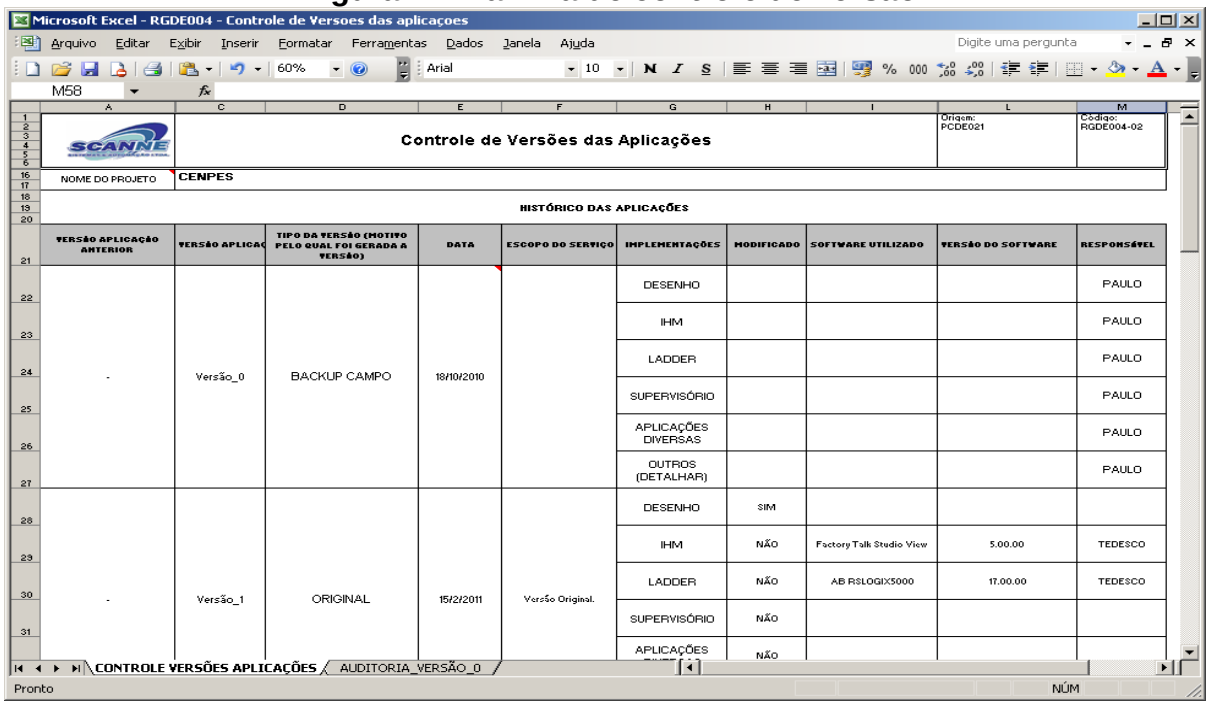
A EIPA/Automação está localizada na sede da UO-SEAL em Aracaju no galpão de automação industrial. Nos Laboratórios de automação são desenvolvidos e testados os Aplicativos e equipamentos utilizados para automação do processo

produtivo da unidade operacional de Sergipe e Alagoas e tem como atribuições: Desenvolver Sistemas de Automação, Gerir Redes de Informações operacionais e Prestar suporte aos Sistemas de Automação.

1.2 Situação Problema

Ao verificar o processo de gestão de configuração dos aplicativos de automação, constatou-se que os métodos atuais de controle das versões dos aplicativos, mantidos em planilhas de Excel, demonstrado na figura 1, não mencionam os milhares de arquivos contidos nas aplicações. Esse problema se deve a falta de uma ferramenta adequada para fazer o controle destes itens de forma automática toda vez que seja criada uma nova versão destes aplicativos.

Figura 1 - Planilha de controle de versão



Fonte: Petrobras

1.3 Objetivos

1.3.1 Objetivo geral

Implementar ferramenta de controle automático dos itens de configuração a cada versão, no processo de ***Gestão de Configuração dos Aplicativos de Automação,*** numa empresa de petróleo do nordeste Brasileiro.

1.3.2 Objetivos específicos

- Pesquisar os softwares existentes no mercado.
- Avaliar os softwares existentes no mercado.
- Utilizar software em projeto piloto.
- Desenvolver planos de ação.
- Implantar a ferramenta de controle de versão no ambiente de automação da empresa.

1.3.3 Justificativa

Na indústria petrolífera a automação industrial possui a função essencial de automatizar o processo produtivo para que se produza de forma segura, contínua e lucrativa. A gestão indevida de seus aplicativos pode levar a recuperação de uma versão equivocada e, com isso, provocar falhas ou danos em equipamentos gerando perda de produção e com prejuízo financeiro para a empresa e também acidentes com pessoas. Sendo assim, é necessário um gerenciamento eficaz e efetivo dos itens de configuração.

2 FUNDAMENTAÇÃO TEÓRICA

Esta etapa do projeto tem como objetivo apresentar e demonstrar os principais fundamentos teóricos necessários ao desenvolvimento desta pesquisa. Inicia-se com o conceito e objetivos da Gestão de configuração de Software e as atividades envolvidas.

2.1 Gestão de Configuração de Software (GCS)

Sommerville (2011, p. 475), se referindo ao gerenciamento de configuração afirma que:

> O gerenciamento de configuração (CM, do inglês configiration managemente) está relacionado com as políticas, processos e ferramentas para o gerenciamento de mudanças dos sistemas de software. Você precisa gerenciar os sistemas em evolução, pois é fácil perder o controle de quais mudanças e versões de componentes foram incorporadas em cada versão de sistema. (SOMMERVILLE, 2011, p. 475).

E prossegue agora se referindo aos procedimentos de GCS:

> Se não temos procedimentos de gerenciamento de configuração efetivos, você pode desperdiçar esforços modificando a versão errada de um sistema, entregá-la para os clientes ou esquecer onde está armazenado o código-fonte do software para uma versão específica do sistema ou componente. (SOMMERVILLE, 2011, p. 475)

Para a Associação para Promoção da Excelência do Software Brasileiro, “O propósito do processo Gerência de Configuração é estabelecer e manter a integridade de todos os produtos de trabalho de um processo ou trabalho e disponibilizá-los a todos os envolvidos.” (SOFTEX, 2012, p. 34).

De acordo com o SWEBOK (2004, p), gerenciar uma configuração significa controlar as mudanças realizadas mantendo a integridade e rastreabilidade do sistema em diferentes pontos no tempo, durante o ciclo de vida do sistema.

Pressmann (2011, p. 514) define a gestão de configuração de software como:

> A Gestão de configuração de software – (SCM), também chamada de gestão de alteração, é o conjunto de atividades destinadas a gerenciar as alterações identificando os artefatos que precisão ser alterados, estabelecendo relações entre eles, definindo o mecanismo para o gerenciamento de diferentes versões destes artefatos, controlando as alterações impostas, e auditando e relatando as alterações feitas. (PRESSMAN, 2011, p. 514).

A Gestão de configuração de acordo com os modelos de melhores práticas:

CobiT 4.1. “Um gerenciamento de configuração eficaz facilita uma maior disponibilidade do sistema, minimiza as questões de produção e soluciona problemas com mais rapidez.” (IT GOVERNANCE INSTITUTE, 2007, p. 135).

CMMI – A gestão de configuração tem como objetivo: “Estabelecer e manter a integridade dos produtos de trabalho através da identificação, controle, verificação e monitoramento constante da situação da configuração.” (FERNANDES; ABREU, 2008, p. 209).

ITIL – O gerenciamento de configuração e de ativo de serviço:

> Abrange a identificação, o registro o controle e a verificação de ativos de serviço e itens de configuração (componentes de TI tais como hardware, software e documentação relacionada), incluindo suas versões, componentes e interfaces, dentro de um repositório centralizado. (FERNANDES; ABREU, 2008, P. 287)

ISO/IEC 20000 – O gerenciamento de configuração nos processos de controle:

> Visa definir e controlar os componentes do serviço e da infra-estrutura relacionada, mantendo atualizadas as informações de sua configuração, englobando atividades de planejamento, identificação dos itens de configuração, controle das alterações e da situação de cada item no seu ciclo de vida (desde a aquisição até o descarte) e auditoria das informações de configuração; (FERNANDES; ABREU, 2008, P. 309)

O processo de gestão de configuração e seus objetivos:

> O processo de gestão de configuração de software define uma série de tarefas que têm quatro objetivos primários: (1) identificar todos os itens que coletivamente definem a configuração do software, (2) gerenciar alterações de um ou mais desses itens, (3) facilitar a construção de diferentes versões de uma aplicação e (4) assegurar que a qualidade do software seja mantida à medida que a configuração evolui com o tempo. (PRESSMAN, 2011, p. 521).

2.1.1 Identificação

“Cada objeto tem um conjunto de características distintas que o identificam de forma única: um nome, uma descrição uma lista de recursos e uma “realização”.” (PRESSMAN, 2011, p. 522).

2.1.2 Itens de configuração

Segundo Molinari (2007, p. 44), num processo de Gerência de

Configuração de Software, um item de configuração é o menor item que será gerenciado e controlado, e todas as suas informações armazenadas.

> Um item de configuração (IC) pode ser um equipamento (hardware), um programa, aplicação ou sistema (software), um documento (manual técnico), ou, ainda, qualquer outro componente que possa ser considerado relevante para o gerenciamento da infra-estrutura de TI. (MAGALHÃES, BRITO, 2007, p. 86)

"Os Itens de configuração são organizados para formar objetos de configuração que possuem um nome e atributos e são "conectados" a outros objetos através de relacionamentos. "(PRESSMAN, 2011, p. 518)

2.1.3 Controle de versão

"O controle de versão combina procedimentos e ferramentas para gerenciar diferentes versões dos objetos de configuração criados durante o processo de software". (PRESSMAN, 2011, p. 523).

> O gerenciamento de versões (VM. Do inglês version management) é o processo de acompanhamento de diferentes versões de componente de software ou itens de configuração e os sistemas em que esses componentes são usados. Ele também envolve a garantia de que as mudanças feitas por diferentes desenvolvedores para essas versões não interfiram umas nas outras. (SOMMERVILLE, 2011, p. 481)

Sommerville (2011, p. 482) salienta a importância da utilização das ferramentas de controla de versão:

> Para dar suporte ao gerenciamento de versões, você sempre deve usar ferramentas de controle de versões (às vezes, chamadas de sistemas de controle de versões ou sistemas de controle de código-fonte). Essas ferramentas identificam, armazenam e controlam o acesso a diferentes versões de componentes. Há disponíveis muitos sistemas de gerenciamento de versões diferentes, incluindo sistemas *open sourse* amplamente usados como CVS e Subversion (PILATO ET AL., 2004; VESPERMAN, 2003). (SOMMERVILLE, 2011, p. 482)

Vários podem ser os recursos oferecidos pelos sistemas de controle de versão:

> 1. *Identificação de versão* e release. Versões gerenciadas recebem identificadores quando são submetidas ao sistema. Normalmente, esses identificadores se baseiam no nome do item de configuração (por exemplo, ButtonManager), seguido por um ou mais números. Por isso o, ButtonManager 1.3 significa a terceira versão na *codeline* 1 do componente ButtonManager. [...].
> *2. Gerenciamento de armazenamento.* Para reduzir o espaço de armazenamento pelas várias versões de componentes que diferem apenas ligeiramente, os sistemas de gerenciamento de versões em geral fornecem recursos de gerenciamento de armazenamento. Em vez de manter ama cópia completa de cada versão o sistema armazena uma lista de diferenças (deltas) entre uma versão e outra. Ao aplicar estas em uma versão-fonte (geralmente, a versão mais recente), uma versão de destino pode ser

> recriada. [...].
> 3. *Registro de histórico de alterações.* Todas as mudanças feitas no código de um sistema ou componente são registradas e listadas. Em alguns sistemas, essas mudanças podem ser usadas para selecionar uma versão específica do sistema. Isso envolve marcar os componentes com palavras-chave que descrevem as mudanças feitas. Em seguida você usa essas marcações para selecionar os componentes que serão incluídos na *baseline.*
> 4. *Desenvolvimento independente.* Desenvolvedores diferentes podem trabalhar no mesmo componente ao mesmo tempo. O sistema de gerenciamento de versões acompanha os componentes que tenham sido verificados para edição e garante que as mudanças feitas em um componente por diferentes desenvolvedores não interfiram.
> 5. *Suporte a projetos.* Um sistema de gerenciamento de versões pode apoiar o desenvolvimento de vários projetos, que compartilham componentes. Em sistemas de suporte a projetos, tal como CVS (VESPERMAN, 2003), é possível fazer *check-in* e *check-out* de todos os arquivos associados a um projeto, em vez de precisar trabalhar com um arquivo ou diretório de cada vez. (SOMMERVILLE, 2011, p. 482).

De acordo com Molinari (2007, p. 169), entre uma revisão e outra somente as diferenças costumam ser armazenadas, isto economiza espaço nos repositórios e para recuperar um determinado arquivo, são analisadas as diferenças e o arquivo é remontado na revisão desejada.

Diferentes versões também agem como backup de maneira que seja possível retornar as versões prévias do software. (MOLINARI, 2008, p. 84).

2.1.4 Repositório

O repositório segundo Pressman (2011, p. 519):

> Nos primeiros tempos da engenharia de software, o repositório era sem dúvida uma pessoa – o programador que tinha de se lembrar da localização de todas as informações relevantes a um projeto de software, aquele que tinha de se lembrar de todas as informações que nunca foram escritas e reconstruir informações perdidas, infelizmente, o uso de uma pessoa como “centro de acumulação e armazenamento” (embora esteja de acordo com a definição *Webster’s Dictonary)* não funciona muito bem. Hoje, o repositório é uma “coisa” – um banco de dados que age como o centro de acumulação e de armazenagem de informações de engenharia de software. O papel da pessoa (o engenheiro de software) é interagir com o repositório usando ferramentas integradas com ele. (PRESSMAN, 2011, p. 519).

De acordo com Pressman (2011, p. 519) o repositório deve apresentar ferramentas que permitam: controlar todas as versões criadas, gerenciar a variedade de relações entre os elementos armazenados, informações de auditoria e manter o controle de configurações que representam marcos de projetos ou versões de produção.

2.1.5 Controle de mudança

Molinari (2007, p. 51) fornece a seguinte definição para controle de mudanças:

“O propósito básico do controle de mudanças é ter o controle total de todo o produto de todo e qualquer pedido (ou requisição) de mudança de um produto e de todas as mudanças implementadas.” (MOLINARI, 2007, p. 51)

Segundo Hirama (2011, p. 131), pode-se considerar que as mudanças serão constantes, pois necessidades acontecem com o tempo e as alterações precisam ser executadas. Controlá-las torna-se inevitável.

Todo resultado dos controles de mudança devem ser documentados e registrados, da mesma forma os relacionamento existente entre os itens de configuração modificados e os pedidos devem ser mantidos. Se tudo for documentado, se torna possível manter a rastreabilidade das alterações, conforme Molinari (2007, p. 85).

Para Pressman (2011, P. 526) o papel do responsável pelo controle de mudanças (CAA- autoridade de controle de alteração) é:

> O papel da CAA é assumir uma visão global, isto é, avaliar o impacto das alterações além da CSI em questão. Como a alteração afetará o hardware? Como a alteração afetará o desempenho? Como a alteração modificará a percepção que o cliente em relação ao produto? Como a alteração afetará a qualidade e a confiabilidade do produto? Essas e muitas outras questões são encaminhadas para o responsável pelo controle das alterações. (PRESSMAN, 2011, P. 526).

2.1.6 Auditoria de configuração

A atividade de auditoria de configuração visa assegurar que as alterações tenham sido implementadas corretamente. A auditoria de configuração busca verificar se existe conformidade entre as características físicas e funcionais do software.

De acordo com Pressman (2011), existem duas maneiras de garantir que as mudanças foram corretamente implementadas: (1) revisões técnicas formais (especifica de que forma a alteração deve ser conduzida tecnicamente) e, (2) auditorias de configuração. A auditoria deve responder às seguintes questões:

As alterações especificadas foram executadas? Alguma mudança

adicional foi incorporada?

Foi realizada a revisão técnica formal para avaliar as correções técnicas?

A alteração seguiu os padrões de desenvolvimento corretamente?

Os atributos da alteração (autor, data de alteração, etc.) nos *Itens de Configuração de Softwares* estão salvos?

Foram atualizados os itens de configuração corretamente?

2.1.7 Relatório de status

O principal objetivo do relatório de status ou análise de status é manter os envolvidos nos projetos informados sobre as alterações ocorridas nos itens de configuração.

O relatório de status disponibiliza a informação necessária para efetivar o gerenciamento do desenvolvimento de um produto e sua posterior manutenção. (MOLINARI, 2008, p. 86).

As informações obtidas pela análise de status podem servir como base para várias medições. Como por exemplo, o número de requisições de mudanças por itens de configuração ou então, o tempo médio necessário para implementar uma mudança, são respostas que a análise de status fornece, de acordo com SWEBOK (2004).

> O relatório de status de configuração às vezes chamado de contabilidade de status é uma tarefa da SCM que responde as seguintes questões: (1) O que aconteceu? (2) Quem o fez? (3) Quando aconteceu?(4) O que mais será afetado? (PRESSMAN, 2011, P. 527).

O responsável pelo repositório de dados deve garantir que os dados e informações extraídos estejam corretos e os relatórios e as consultas totalmente automatizadas, de acordo com Molinari (2007, p. 54).

2.2 Software livre

Segundo Falcão (2005), através da GNU GPL (General Public License) foram estabelecidos os quatro pilares básicos do software livre.

> Esses pilares consistem em quatro liberdades fundamentais que definem se um software é livre ou não. São elas:
> a) A liberdade de executar o programa, para qualquer propósito.
> b) A liberdade de estudar como o programa funciona, e de adaptá-lo às suas necessidades. O acesso ao código-fonte é uma condição prévia para o

> exercício dessa liberdade.
> c) A liberdade de redistribuir cópias, de modo que você possa auxiliar outras pessoas.
> d) A liberdade de aperfeiçoar o programa e distribuir esses aperfeiçoamentos para o público, de modo a beneficiar toda a comunidade. O acesso ao código-fonte é também uma condição prévia para o exercício dessa liberdade. (FALCÃO; et al, 2005, p. 7)

2.3 Aprendizagem colaborativa e participação ativa

Com base no estudo realizado por Falcão et. al. (2005) para o Instituto Nacional de Tecnologia, o aprendizado obtido de maneira colaborativa proporciona o enriquecimento técnico e favorece a troca de informações de forma sincrônica ou assincrônica. Este modelo de cooperação presentes em comunidades de Softwares é baseado em uma relação solidária que difere da relação comercial, pois é dada aos participantes a chance de modificar a criação de outro integrante que tenha inventado qualquer programa, englobando assim a participação da sociedade no desenvolvimento de tecnologias para seu próprio benefício, porém sem intenção lucrativa.

“O conhecimento não é um conjunto estático de informações, mas um processo dinâmico de revisão e aperfeiçoamento desses conteúdos.” (FALCÃO; et. al, 2005, P. 96)

3 METODOLOGIA

3.1 Método

Esta é uma pesquisa na área da engenharia. Quanto aos fins trata-se de uma pesquisa aplicada. “Pesquisas voltadas à aquisição de conhecimentos com vistas à aplicação situação específica”. (GIL, 2010, p. 27).

Devido aos métodos empregados se classifica como um estudo de caso.

Para Gil (2010, p. 37) o estudo de caso “consiste no estudo profundo e exaustivo de um ou poucos objetos, de maneira que permita seu amplo e detalhado conhecimento”.

Quanto aos objetivos trata-se de uma pesquisa exploratória descritiva, porque busca uma solução para resolver um problema de automação de versão de aplicativos de uma empresa de petróleo.

3.2 Universo e Amostra

O universo de estudo são as ferramentas disponíveis no mercado de software de aplicativos de controle de versão, encontrados a partir de levantamento realizado na internet.

As amostras foram selecionadas pelos critérios de intencionalidade, pois existe a preferência por qualidade nas amostras.

> Uma amostra intencional, em que os indivíduos são selecionados com base em certas características tidas como relevantes pelos pesquisadores e participantes, mostra-se mais adequada para obtenção de dados numa pesquisa-ação. (GIL, 2010, p.153)

3.3 Coleta e Análise de Dados

A coleta dos dados foi realizada inicialmente através de informações disponível nos sites dos desenvolvedores das ferramentas de controle de versão.

Os dados foram analisados de acordo com o com representatividade da ferramenta de controle de versão no grupo investigado e no fundamento teórico, foi

selecionada uma das ferramentas de controle de versão para um projeto piloto. Como se trata de uma pesquisa aplicada, foi elaborado um plano de implantação da ferramenta de controle de versão escolhida e o resultado prático está demonstrado na implementação da ferramenta de controle de versão.

4 ANÁLISE DE RESULTADOS

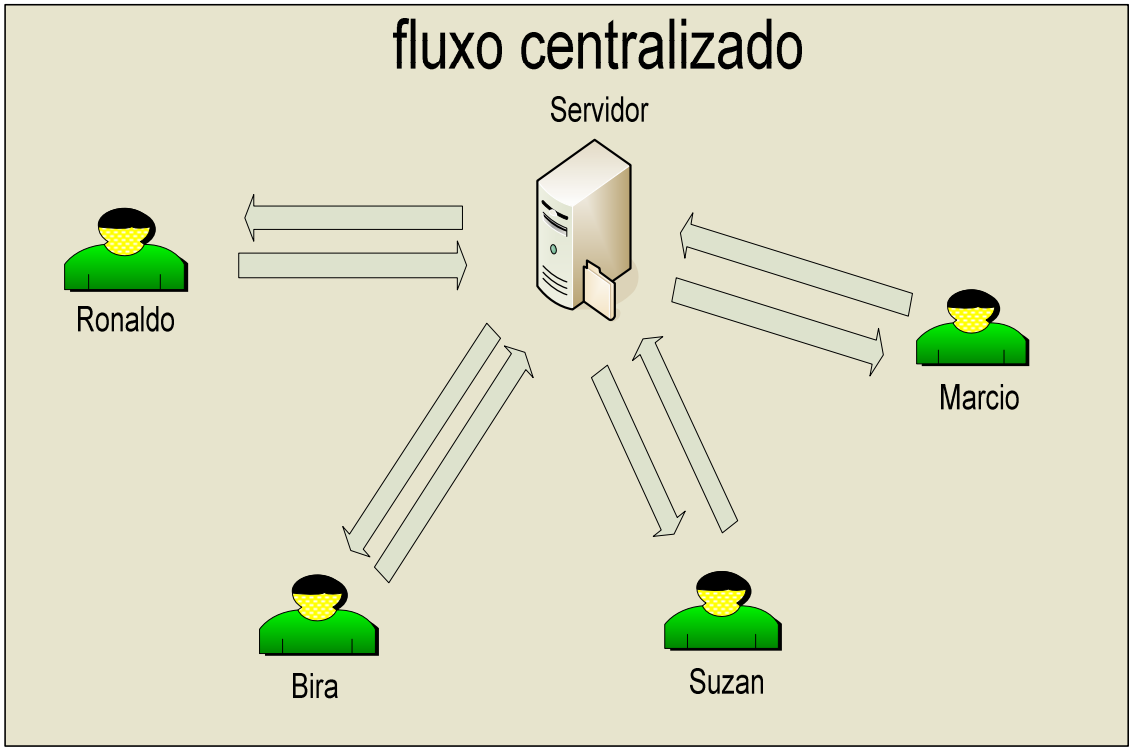
4.1 Pesquisar os Softwares de Controle de Versão Existentes no Mercado

Atualmente existem vários softwares de controle de versão no mercado e são classificados em dois tipos: distribuído e centralizado. Neste trabalho, foram pesquisados os dois tipos.

4.1.1 Centralizado

Neste tipo de Controle de Versão, as informações referentes aos arquivos estão um repositório único, acessível (dentro de um determinado contexto de políticas de segurança específica) a todos os desenvolvedores de um projeto, utilizando uma arquitetura tipo Cliente/Servidor, ou seja, fluxo centralizado, demonstrado na figura 2.

Figura 2 - Fluxo centralizado



Fonte: Autor

Em um sistema centralizado, cada desenvolvedor mantém uma cópia de

trabalho com os arquivos contidos no repositório, além de uma pasta com metadados referentes a estes arquivos, que são utilizados para comparar com as informações existentes no repositório. Ao modificar um arquivo, o desenvolvedor pode atualizar o repositório, através de uma ferramenta cliente, tornando assim publicas suas alterações, este processo é conhecido como "commit".

4.1.1.1 ferramentas de controle de versão centralizado

Visual Source Safe (VSS)

De acordo com a Microsoft (2013), o Visual SourceSafe, é um sistema de controle versão desenvolvido pela Microsoft, permite edição colaborativa e compartilhamento de dados entre vários desenvolvedores dentro de uma organização permite trabalhar em várias versões de um projeto ao mesmo tempo. Projetado para proteger contra perdas ecidentais de arquivos, rastrear revisões de projetos inteiros, permite trabalhar com mesclagem de arquivos.

Concurrent Version System (CVS)

De acordo com Pressman (2011) o CVS é um sistema de controle de versão amplamente utilizado, permite trabalhar com os arquivos organizados em um repositório, mantém todas as versões armazenando apenas as diferenças entre as versões progressivas do original. Protege os arquivos contra alterações simultâneas. O CVS mescla as alterações de diversos desenvolvedores.

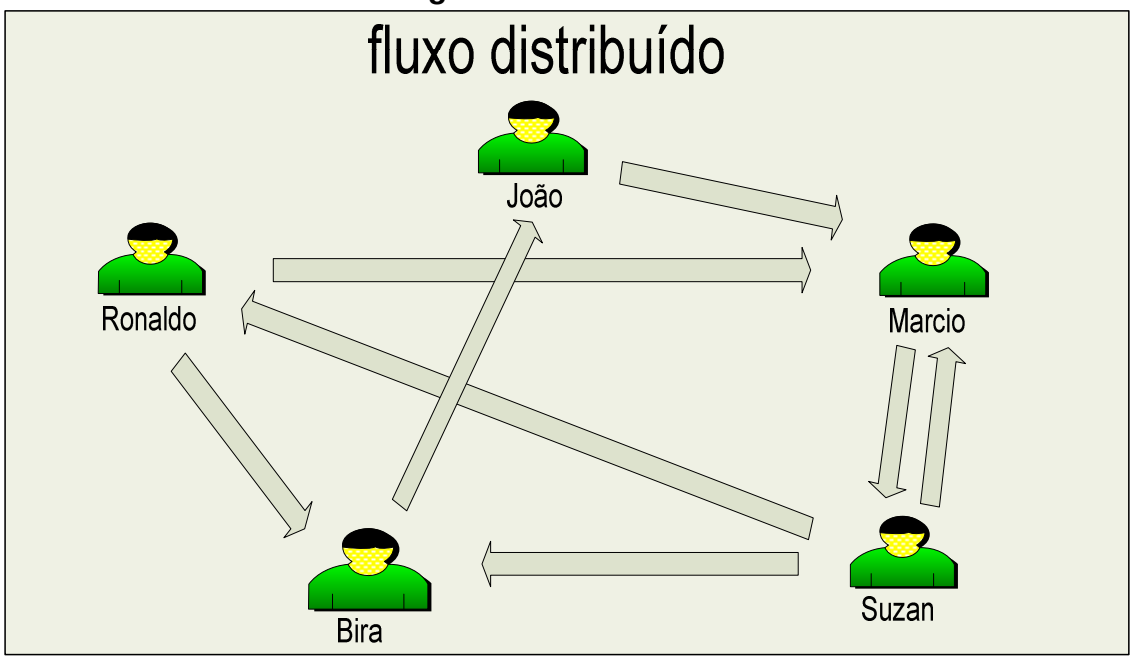
Subversion

De acordo com o desenvolvedor atual a Apache Software Foundation (2013), o Subversion é um sistema de controle de versão de código aberto, é um sistema de uso geral que permite gerenciar qualquer coleção de arquivos e diretórios e as alterações feitas a eles ao longo do tempo. Disponibiliza commits atômicos, ou seja, os commits só têm efeito se todo o commit for bem sucedido.

4.1.2 Distribuído

Neste tipo de sistema cada desenvolvedor possui um repositório independente, podendo trabalhar sem acesso a uma rede ou servidor central. Os repositórios são completos com histórico de todas as versões, possibilita vários tipos de fluxos entre os repositórios, conforme pode ser visto na Figura 3.

Figura 3 - Fluxo distribuído



Fonte: autor

Nos sistemas distribuídos a maior parte das operações é realizada no próprio repositório, tornando o sistema mais rápido e ágil que o sistema centralizado.

4.1.2.1 ferramentas de sistema de controle de versão distribuído

Mercurial

Segundo a Comunidade Mercurial (2013), o Mercurial foi criado por Matt Mackall, e tem licença livre de código aberto, permite verificar diferenças entre versões de desenvolvimento, fornece verificação de integridade de dados do repositório, os dados são armazenados por acréscimo, possui um repositório completo de verificação, tem uma facilidade de backup e inclui uma interface web integrada.

Bazaar

Segundo a Comunidade Bazaar (2013), Bazaar é um software livre patrocinado pela Canonical, é um sistema de controle de versão que ajuda a controlar o histórico do projeto ao longo do tempo, permite trabalhar *offline* e possui alta eficiência de armazenamento, se adapta a qualquer fluxo de trabalho centralizado ou distribuído.

Git

De acordo com a Chacon (2013), o Projeto foi desenvolvido por Linus

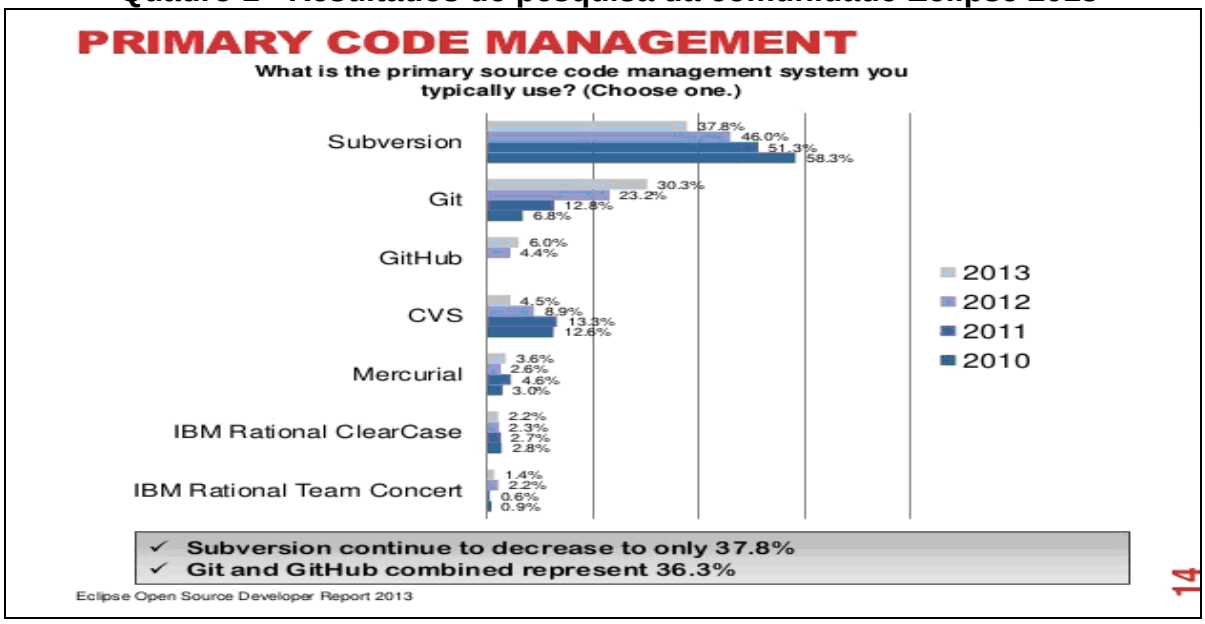
Torvalds (criador do Linux, núcleo do sistema operacional GNU/Linux), o Git é uma ferramenta livre de código aberto, é rápido e eficiente em projetos pequenos ou grandes, cada clone de um repositório é um espelho do repositório original. Seu clone contém o histórico completo das mudanças e revisões, e não está vinculado a um servidor central.

4.2 Avaliar os Softwares Existentes no Mercado

Os critérios usados para avaliação dos softwares pesquisados foram: massa critica de utilização em projetos, que determina a aceitabilidade e continuidade do software no mercado; compatibilidade com sistemas operacionais; tipo de licença (licença livre ou proprietária); adaptação ao processo de automação.

De acordo com a pesquisa publicada em julho 2013 pela fundação eclipse, que foi realizada no período de 12 abril a 10 de maio de 2013, entre os membros da comunidade Eclipse e, onde 920 membros responderam a pesquisa. Pode-se observar no quadro 1 que duas das ferramentas pesquisadas tiveram destaque; o Subversion e o Git, que nos últimos quatro anos demonstrou uma aceitação de 37,8% e 36,3% respectivamente.

Quadro 1 - Resultados de pesquisa da comunidade Eclipse 2013

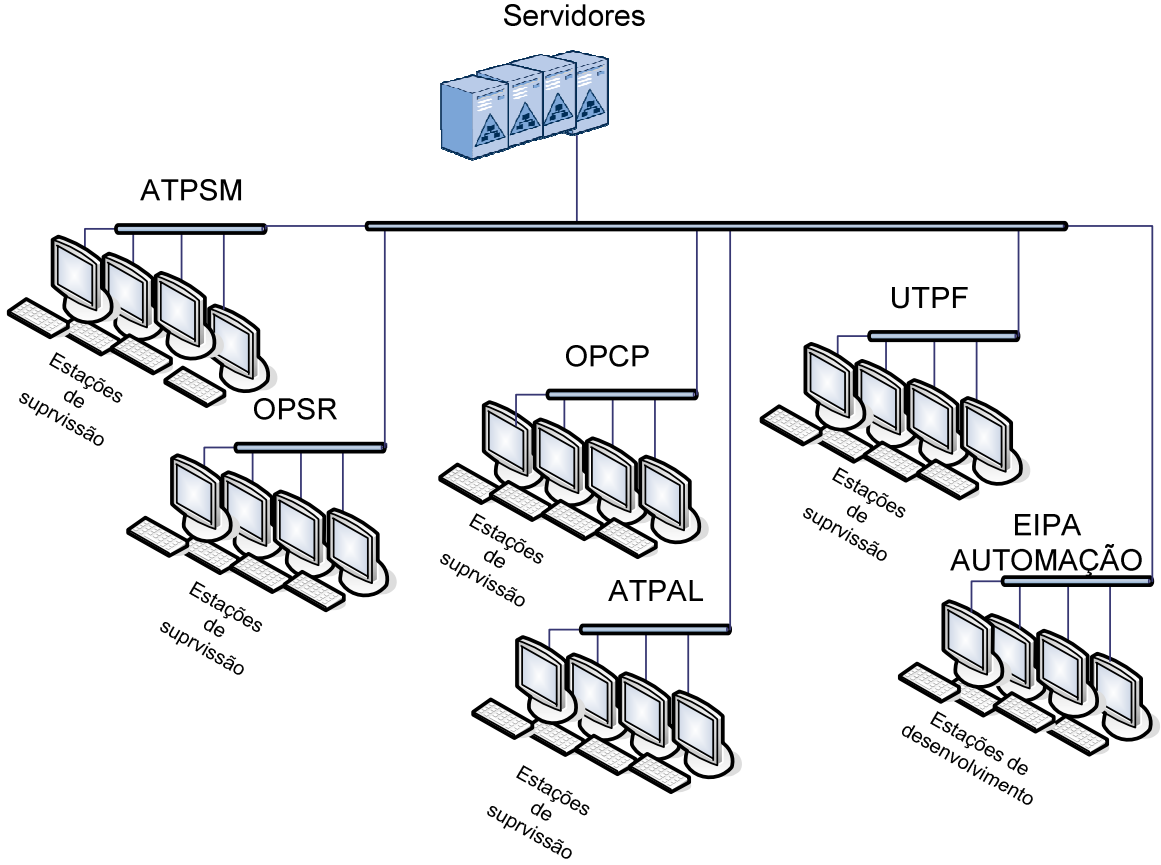


Fonte: Comunidade Eclipse.

Diante do resultado de critério massa critica as ferramentas selecionadas foram o subversion e o Git. Então se partiu para análise de compatibilidade de sistema operacional ou plataforma de trabalho, porém ambos são compatíveis com

ambiente Windows e Linux. Logo o critério decisivo foi o modelo da arquitetura da rede de automação da empresa pesquisada. Na rede de automação estão todos os computadores de desenvolvimento da EIPA automação e os computadores das estações de supervisão dos Ativos de Produção (ATPAL, OPCP, OPSR, ATPSM, UTPF). Esta rede é segregada de outras redes da empresa, por motivo segurança e integridade da mesma, sendo permitido o acesso apenas aos servidores da empresa, como se verifica na figura 4.

Figura 4 - Arquitetura de automação (simplificada)



Fonte: Autor

A utilização de uma arquitetura tipo Cliente/Servidor esta mais apropriada, e seguindo as orientações estabelecidas no modelo de melhores praticas de TI o CobiT 4.1, onde no processo gerenciar a configuração no item DS9.1, define:

> “Estabelecer uma ferramenta de suporte e um repositório central para conter todas as informações relevantes sobre os itens de configuração. Monitorar e registrar todos os bens e as mudanças ocorridas neles. Manter um perfil básico de itens de configuração de todo sistema e serviço como um ponto de verificação seguro para eventual retorno após as mudanças.” (IT GOVERNANCE INSTITUTE, 2007, p. 136.)

Decidiu-se por utilizar a ferramenta de controle de versão centralizada, o Subversion.

4.2.1 O Subversion

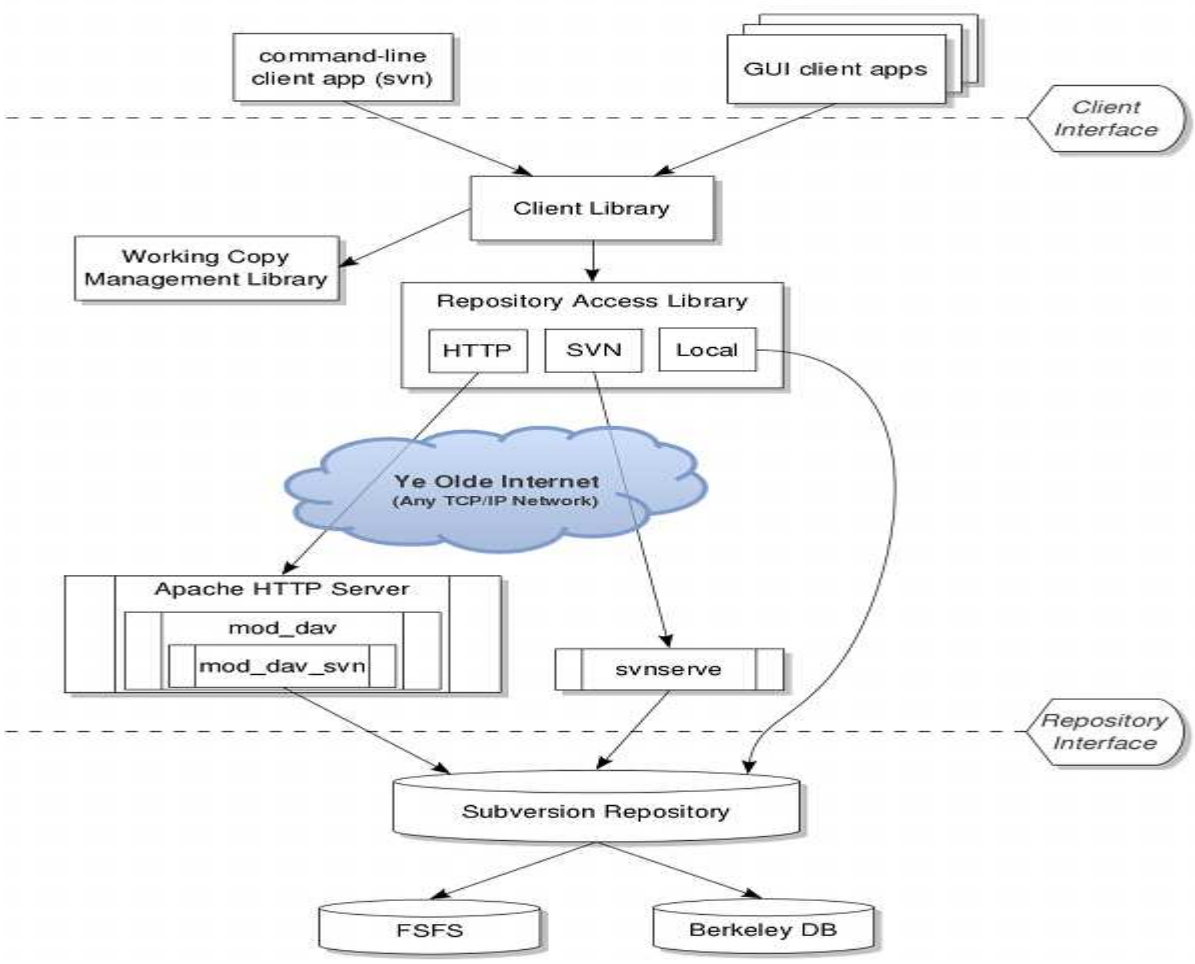
Subversion é uma ferramenta de controle de versão conforme a definição dos seus criadores:

> Subversion é um sistema de controle de versão livre/open-source. Isto é, o Subversion gerencia arquivos e diretórios, e as modificações feitas neles ao longo do tempo. Isto permite que você recupere versões antigas de seus dados, ou que examine o histórico de suas alterações. Devido a isso, muitas pessoas tratam um sistema de controle de versão como uma espécie de "máquina do tempo". (COLLINS-SUSSMAN; FITZPATRICK; PILATO, 2007, p.1)

4.2.1.1 arquitetura do Subversion

A arquitetura do Subsersion está demonstrada na figura 5, segue uma filosofia de modelo do tipo cliente/servidor.

Figura 5 - A arquitetura do Subversion



Fonte: Collins-Sussman; Fitzpatrick; Pilato (2007, p.XX)

De acordo com, Collins-Sussman; Fitzpatrick; Pilato (2007, p.XX) em um

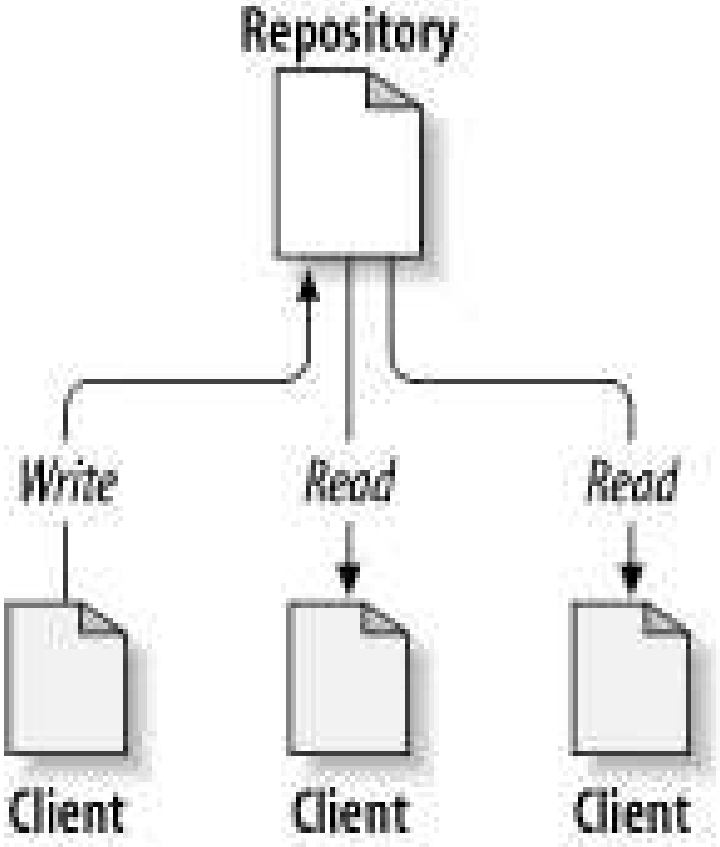
extremo está o repositório Subversion que mantém todos os dados versionados e, no outro extremo está o programa cliente Subversion, que controla localmente partes desses dados versionados. Entre os dois extremos estão às várias rotas disponíveis para acesso ao repositório, que podem ser através das redes de computadores e por meio de servidores de rede, ou diretamente o repositório.

> Subversion é construído usando uma biblioteca de portabilidade chamada de APR (Apache Portable Runtime). Por isso, o Subversion roda em qualquer sistema operacional que o servidor Apache roda: Windows, Linux, BSD, Mac OS X, Netware e outros. (DIAS, 2011, p.4)

4.2.1.2 o repositório

Segundo Collins-Sussman; Fitzpatrick; Pilato (2007, p. 1), o repositório do Subversion armazena informação em forma de uma árvore de arquivos, registra toda alteração ocorrida em seus diretórios e mudanças em cada um de seus arquivos, seja de adição, eliminação ou reorganização. Qualquer cliente que se conecta ao repositório pode ler ou escrever nestes arquivos. Quando os dados são gravados, a informação fica disponível para outros, a figura 6 mostra esta arquitetura

Figura 6 - Sistema cliente/servidor



Fonte: Collins-Sussman; Fitzpatrick; Pilato (2007, p.1)

4.2.1.3 cópias de trabalho

Segundo Collins-Sussman; Fitzpatrick; Pilato (2007, p. 7) No seu computador local, uma cópia de trabalho do Subversion é uma árvore de diretórios comum, contendo múltiplos arquivos, e também um subdiretório extra chamado .svn, criados e mantidos pelo Subversion, conhecido como o diretório administrativo da cópia de trabalho local. Os arquivos deste diretório mantêm o Subversion informado sobre os arquivos que possuem alterações não-publicadas, e quais estão desatualizados em relação ao trabalho dos outros.

O Subversion disponibiliza alguns comandos básicos para interação da copia de trabalho com o repositório, conforme Quadro 2.

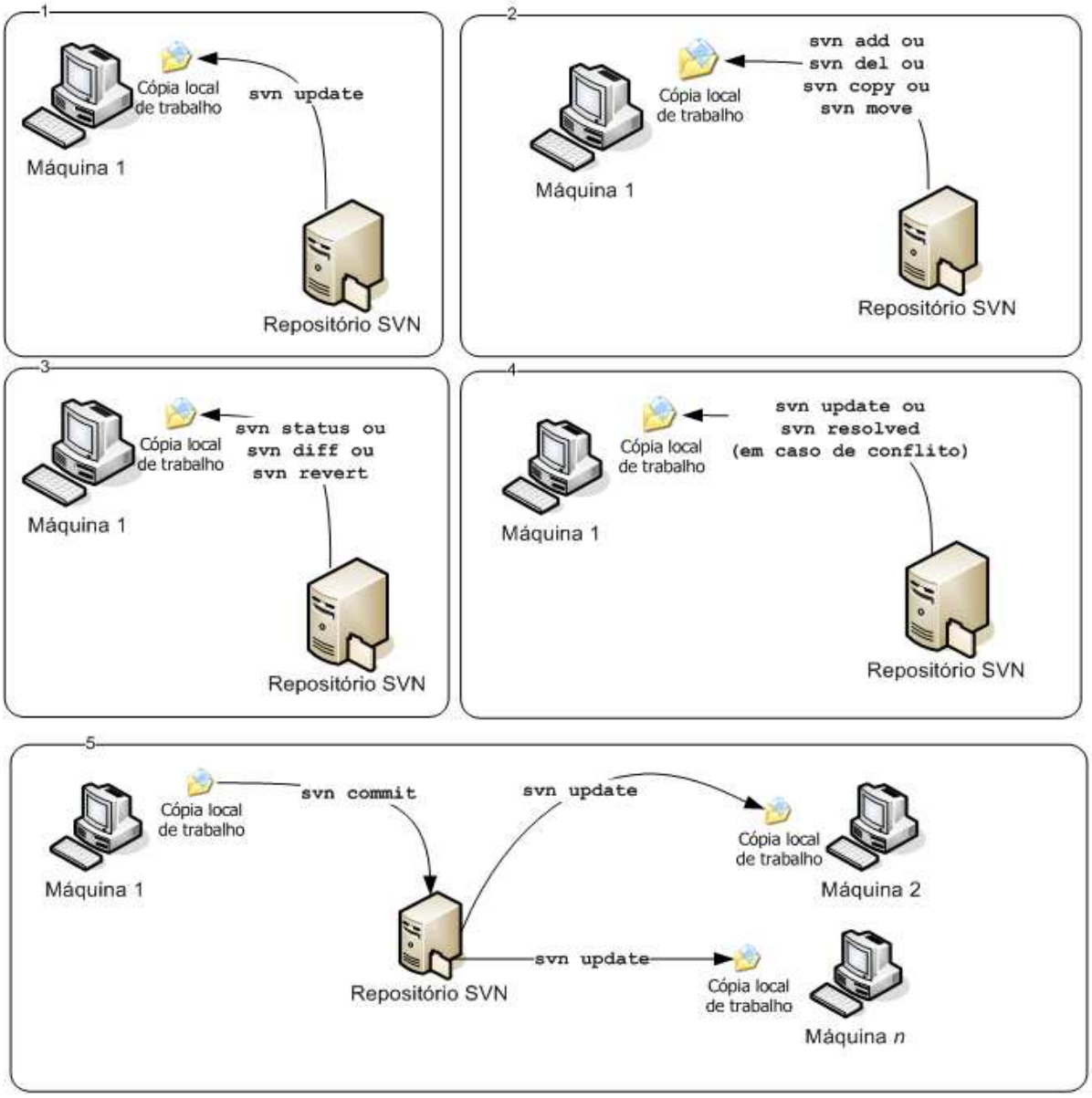
Quadro 2 - Lista de Comandos

Comando	Atalho	Descrição
checkout	co	Cria uma cópia de trabalho a partir do repositório
update	up	Atualiza a cópia local com a versão mais atual no repositório
commit	ci	Envia as alterações da cópia de trabalho para o repositório
revert		Reverte arquivo(s)/diretório(s) ao estado de sua última atualização junto ao repositório.
status	st	Informa qual a situação da cópia de trabalho
add		Adiciona conteúdo à cópia local – e após um commit, ao repositório
remove	rm	Remove conteúdo do repositório ou da cópia de trabalho
info		Exibe informações sobre o repositório, cópia de trabalho ou conteúdo
log		Exibe histórico sobre conteúdo ou repositório
diff		Exibe diferenças na cópia de trabalho ou em arquivos listados

Fonte: SIMPLES CONSULTORIA

De acordo com Rentes (2009, p. 15). O Subversion possui muitas funções, opções, possibilidades de uso, mas para a utilização diária só se irá usar um sub-conjunto limitado das funcionalidades. O ciclo de trabalho é tipicamente o apresentado na Figura 7.

Figura 7 - Ciclo normal de trabalho



Fonte: RENTES

Segundo Rentes (2009, p. 15), a instalação do Subversion em Windows pode ser feita instalando-se os binários do cliente/servidor de SVN e usar apenas o subversion numa shell DOS, o que não é muito interessante para os utilizadores do Windows, ou instalar um cliente gráfico.

4.2.2 Cliente gráfico TortoiseSVN

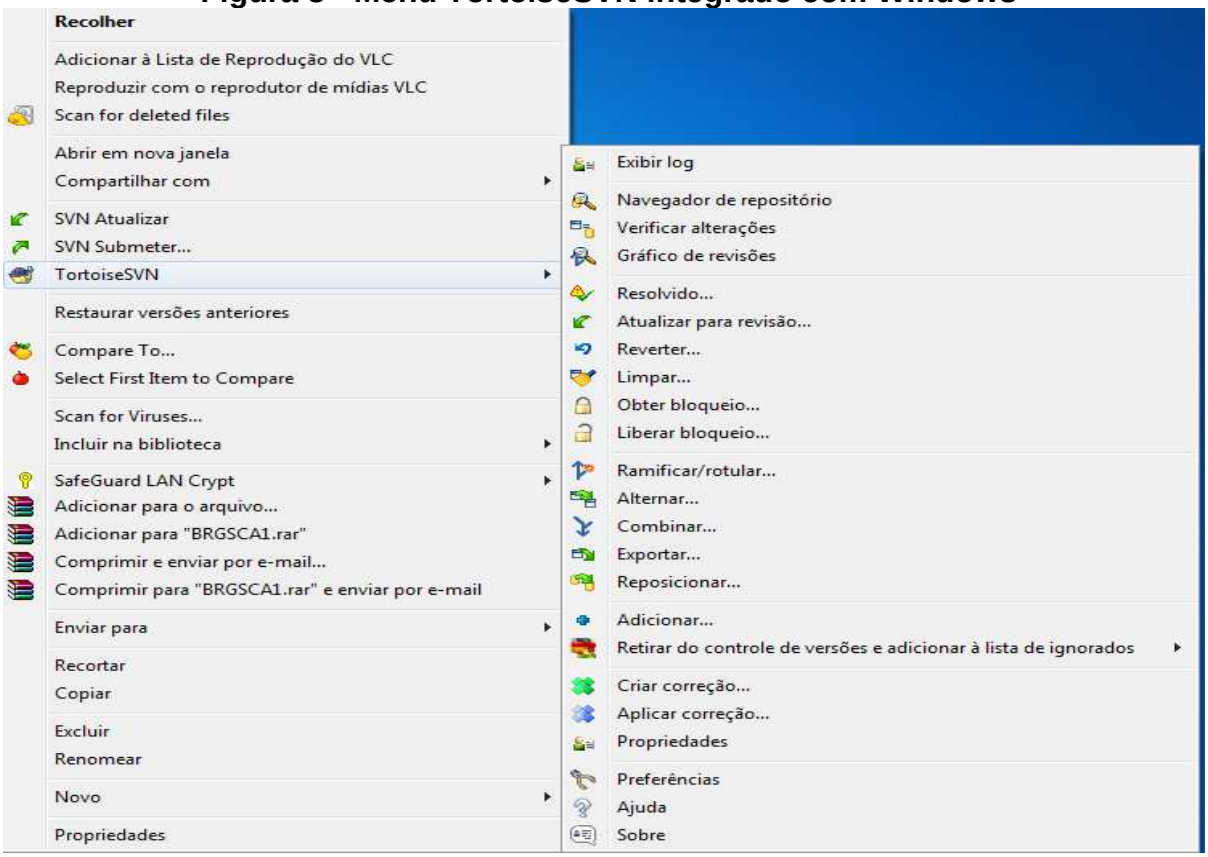
Segundo Küng; Onken; Large (2013, p.xii), TortoiseSVN é um Programa Cliente Subversion para ser usado na plataforma Windows, é um projeto Open Source desenvolvido sob a licença GNU General Public License (GPL), é gratuito

para baixar e para usar, para fins pessoais ou comerciais, em qualquer número de computadores e, em seu pacote de idiomas encontra-se a versão em Português do Brasil.

4.2.2.1 interface integrada ao Windows

O TortoiseSVN integra-se ao shell do Windows, ou seja, Explorer. Basta Clicar com o botão direito numa pasta do Explorer para ver algumas entradas novas no menu de contexto como demonstrado na figura 8, e segundo S. Küng; L. Onken; S. Large, 2013, todos os comandos do Subversion estão disponíveis nos menus.

Figura 8 - Menu TortoiseSVN integrado com Windows



Fonte: Autor

4.2.2.2 sobreposição de ícones

A situação de cada arquivo e diretório controlado na copia de trabalho local, é indicado por uma pequena sobreposição de ícones conforme figura 9. O que permite a visualização rápida da situação de seus arquivos e diretórios.

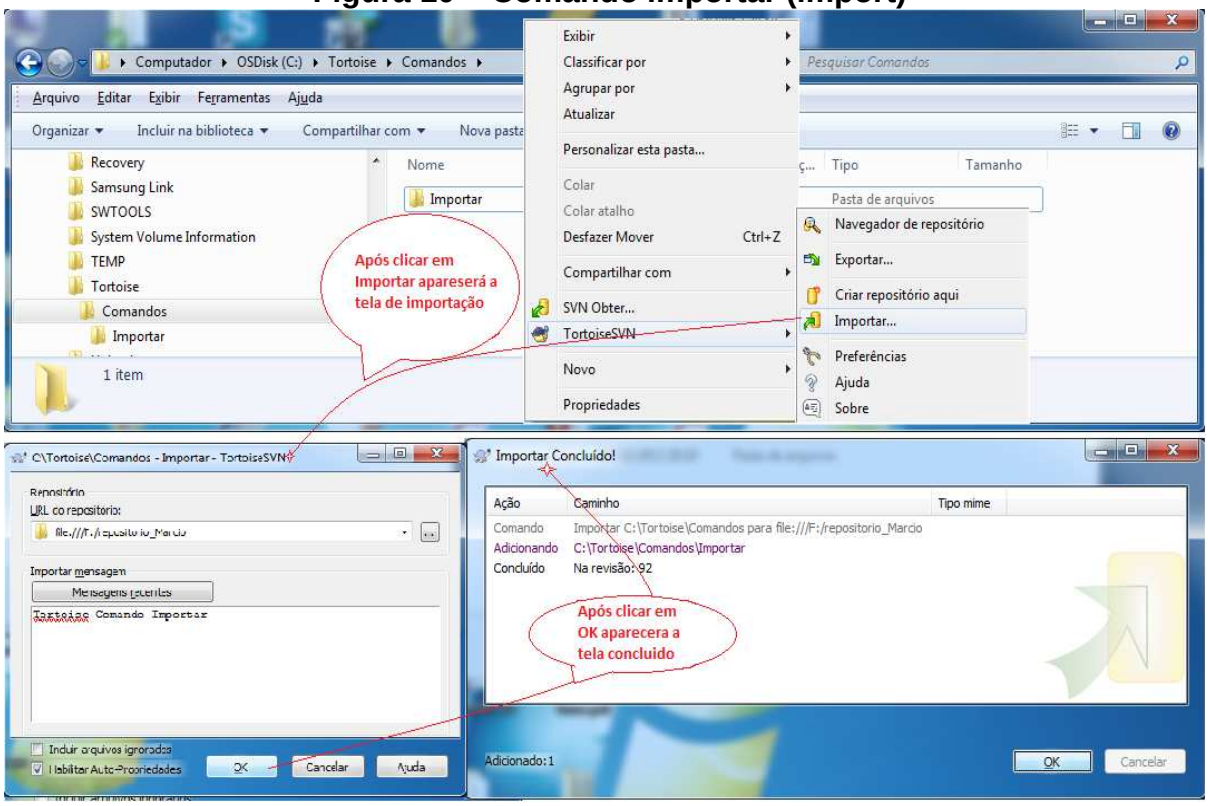
Figura 9 - Icones de sobreposição do TortoiseSVN

Fonte: KÜNG; ONKEN; LARGE.

4.2.2.3 operações básicas do TortoiseSVN

Após ter-se criado um repositório e instalado o tortoiseSVN no seu computador, pode-se começar importando um diretório de arquivos para o repositório conforme pode ser visto na figura 10.

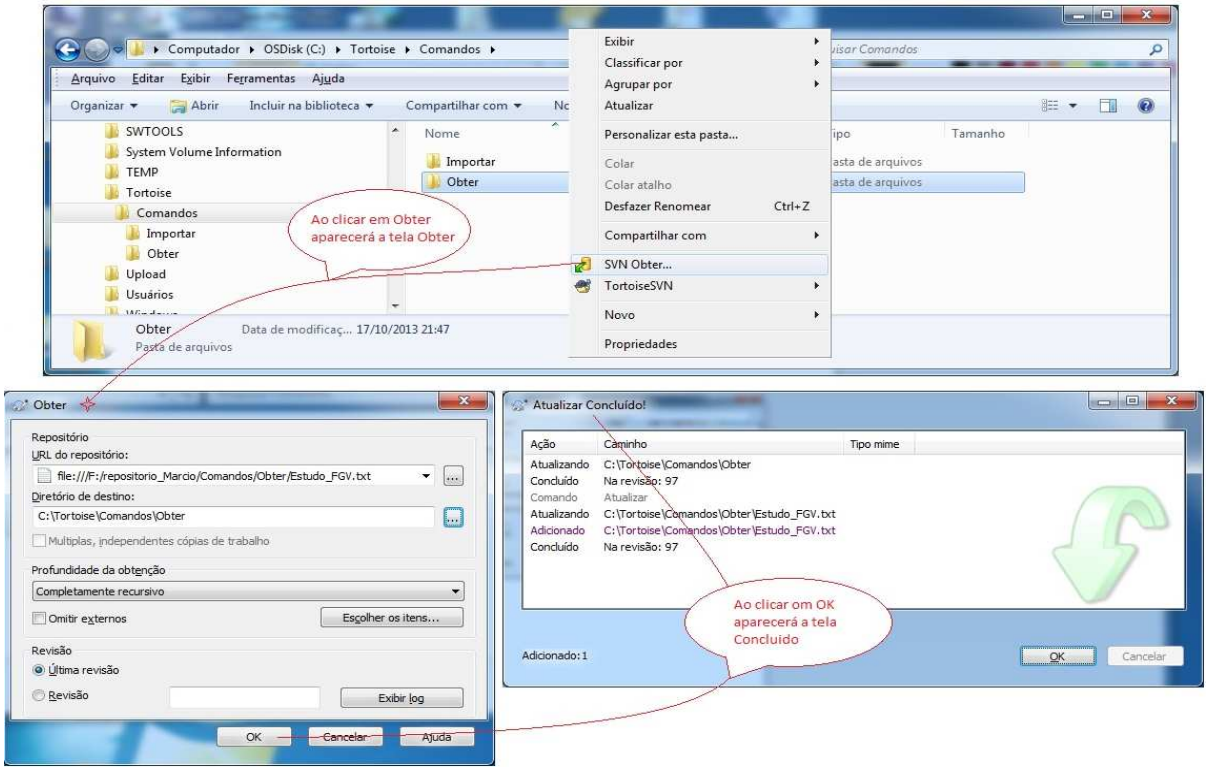
Figura 10 – Comando importar (import)



Fonte: Autor

Para cria uma cópia de trabalho a partir do repositório foi utilizado o comando obter, o qual pode ser visualizado na figura 11.

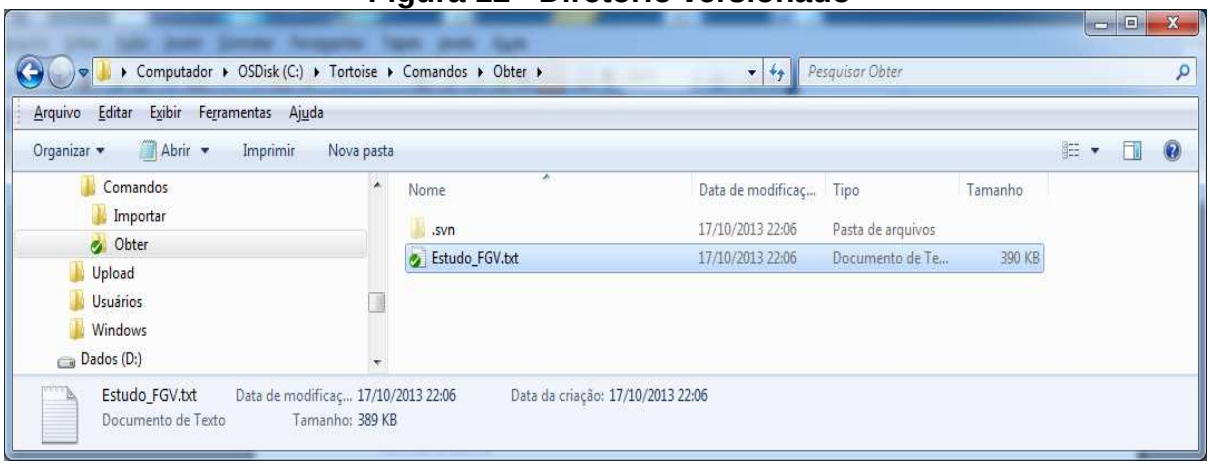
Figura 11 - Comando obter (checkout)



Fonte: Autor

É possível observar na figura 12, que a pasta do diretório obter esta com o ícone de normal sobreposto, e dentro da pasta encontram-se um subdiretório .svn criado pelo subversion e conhecido como diretório administrativo da cópia de trabalho local. Também pode ser encontrado o arquivo obtido do repositório com o ícone normal sobreposto.

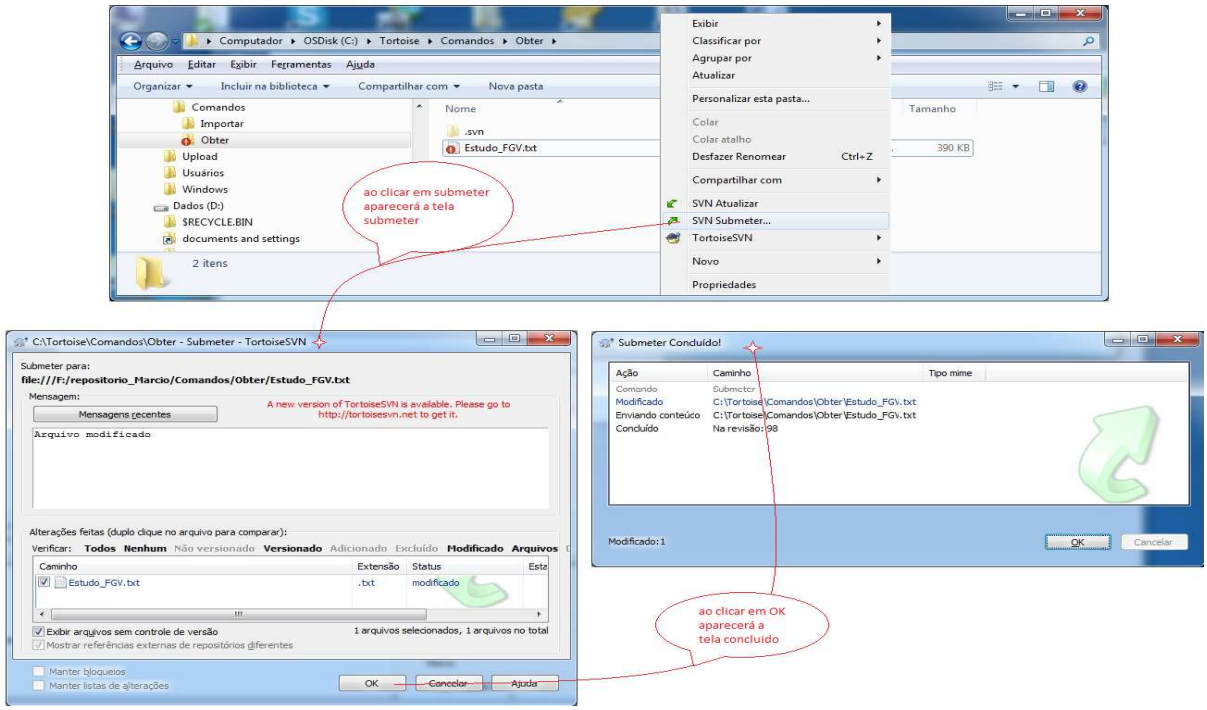
Figura 12 - Diretório versionado



Fonte: Autor

Alterando-se o conteúdo do arquivo versionado, o ícone do mesmo muda para modificado, e para que o repositório reconheça estas alterações, o arquivo deve ser enviado ao repositório utilizado o comando Submeter, conforme a figura 13.

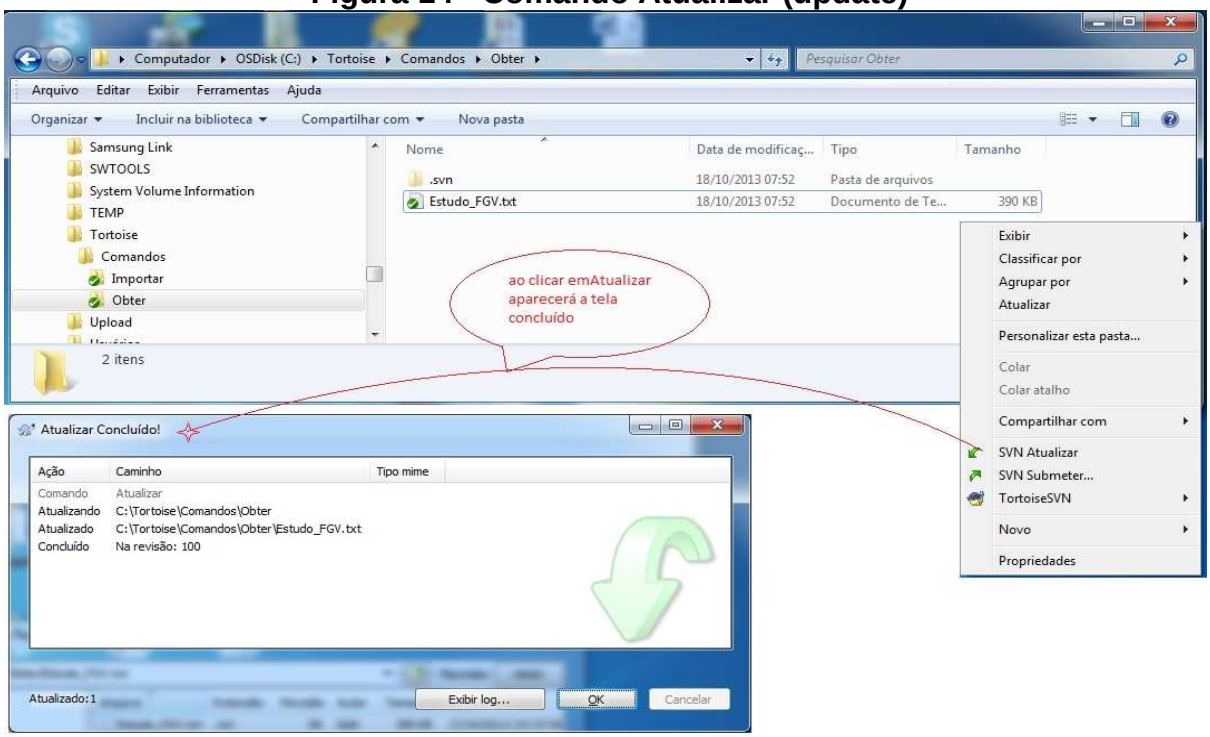
Figura 13 - Comando Submeter (commit)



Fonte: Autor

Periodicamente deve-se atualizar a cópia de trabalho local para garantir que as alterações realizadas por outros no repositório sejam incorporadas, utilizando-se do comando Atualizar do TortoiseSVN. Conforme demonstrado na figura 14.

Figura 14 - Comando Atualizar (update)

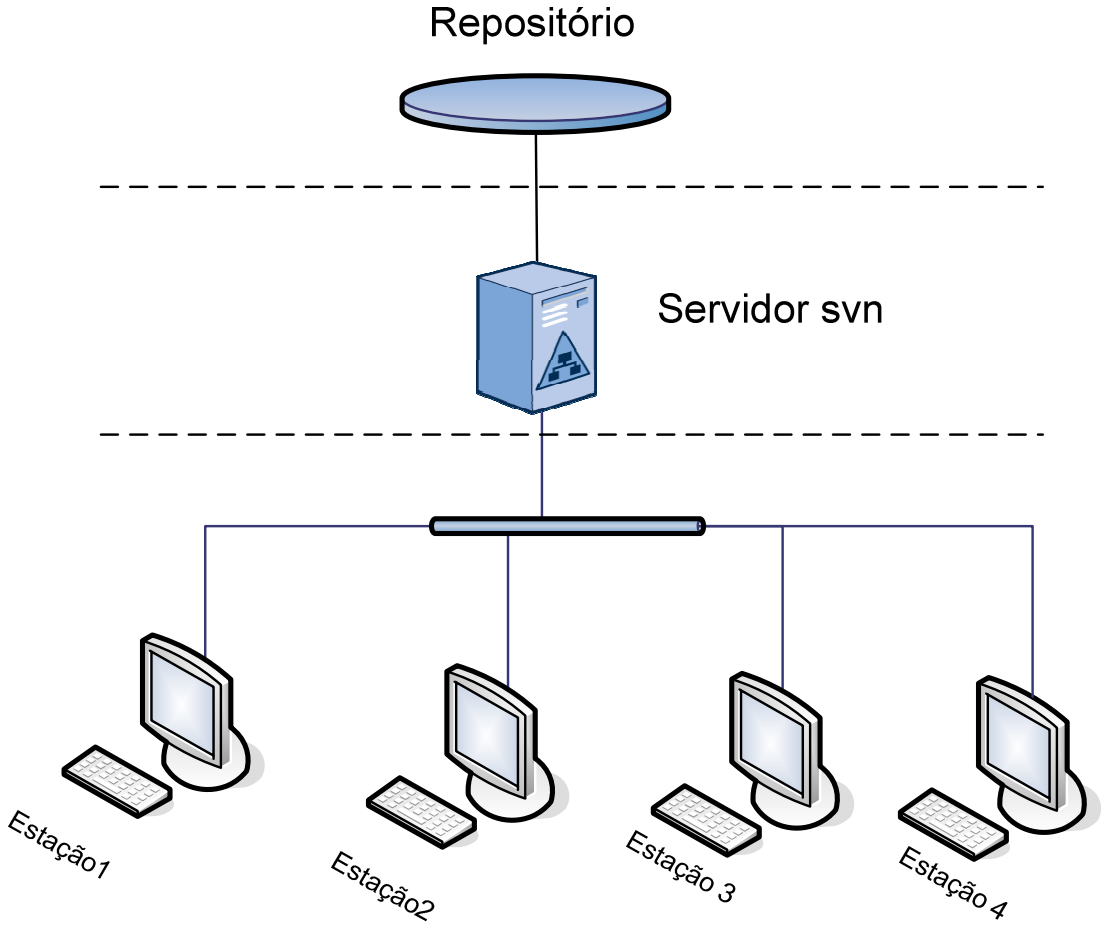


Fonte: Autor

4.3 Utilizar Software em Projeto Piloto.

Para verificar a viabilidade do uso das ferramentas (Subversion e TortoiseSVN), foi concebido um projeto piloto. Onde será avaliado o funcionamento das ferramentas no ambiente de automação. O projeto piloto foi concebido em ambiente controlado simulando o ambiente de automação, como demonstrado na figura 15.

Figura 15 - Ambiente de automação



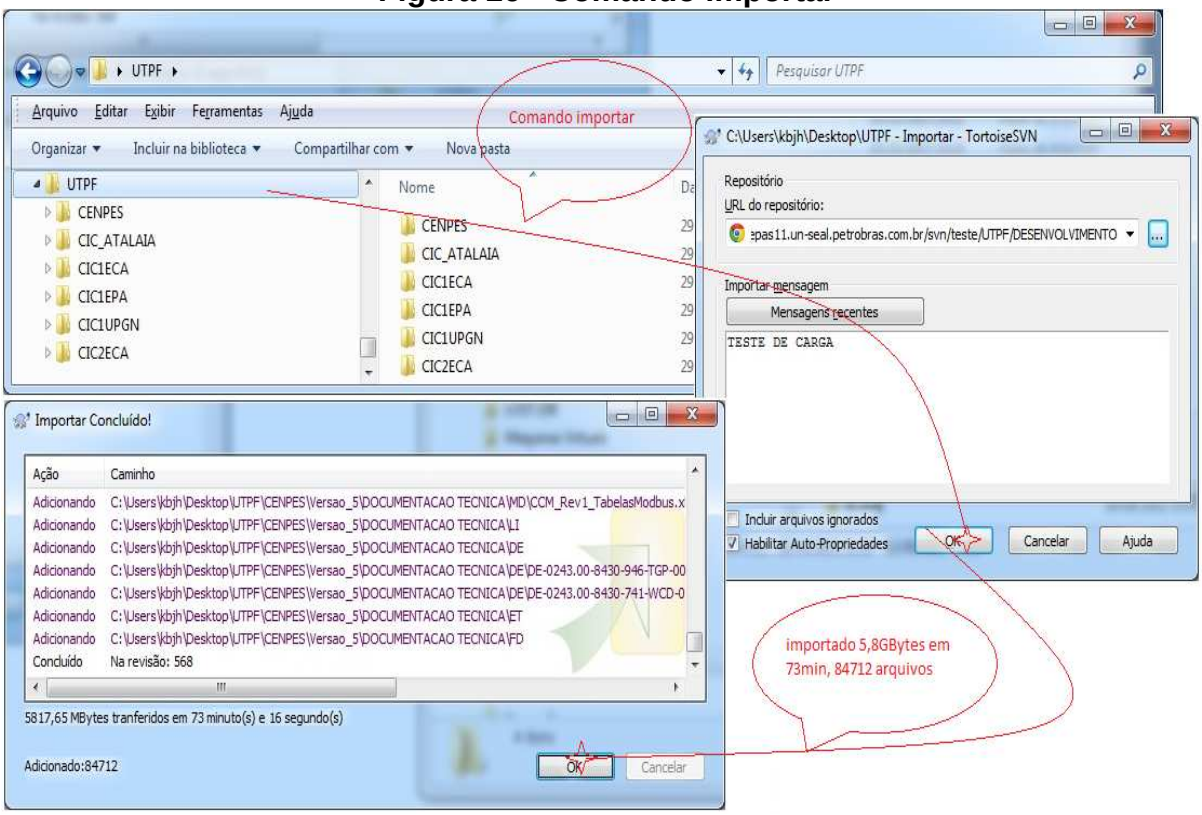
Fonte: Autor

Para isso foi necessário criar um ambiente para projeto piloto com, um computador servidor visualSVN e repositório, e outras máquinas com cliente TortoiseSVN salientando que todas estão rodando numa plataforma WindowsXP.

Os testes iniciaram enviando ao servidor uma carga de dados de 23,17 Gbytes de informações distribuído entre as estações, com objetivo de verificar o tempo necessário de envio de dados para o servidor. A carga foi realizada para o

repositório utilizando a ferramenta cliente TortoisSVN com o comando importar, conforme apresentado na figura 16.

Figura 16 - Comando importar



Fonte: Autor.

Pode ser observado que o teste realizado na estação 4, para uma carga de 4,97GB, levou um tempo de 61 minutos, já para estação 2 de 7,04 gastou um tempo de 90 minutos para envio dos dados. Demonstrando uma linearidade entre a carga e o tempo, conforme demonstrados no quadro 3.

Quadro 3 - Teste de carga

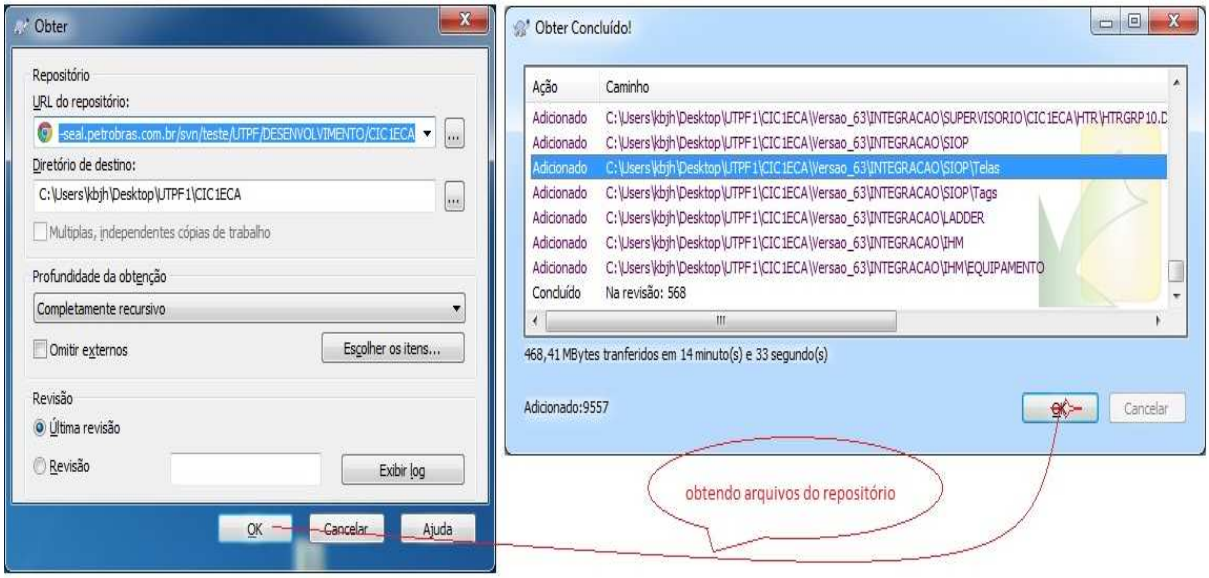
Computadores	N°DIRETÓRIOS	Total de arquivos	Carga em GB	Tempo em min.
Estação 1	6	84.712	5,81	73
Estação 2	11	39.466	7,04	90
Estação 3	10	51.416	5,35	69
Estação 4	8	47.978	4,97	61
TOTAL	35	223.572	23,17	293

Fonte: Autor.

As ferramentas demonstraram um bom desempenho quanto aos tempos de importação em torno de 5 horas para importar uma carga de 23,17GB.

Para obter uma copia de trabalho foi utilizado o comando obter para baixar as aplicações para as estações 1, 2, 3 e 4, demonstrado na figura 17.

Figura 17 - Comando obter (checkout)



Fonte: Autor.

Quanto ao tempo para criar uma cópia de trabalho, na estação 3 foi de 11 minutos para uma carga 355,35MB, já para a estação 4 tem-se um tempo de 27 minutos para uma carga de 854,65MB, isso é considerado normal devido à criação por parte do Subversion de um subdiretório administrativo .svn, que é utilizado para controle dos arquivos da copia de trabalho, que estão demonstrados na tabela 3.

Quadro 4 - Teste com o comando obter

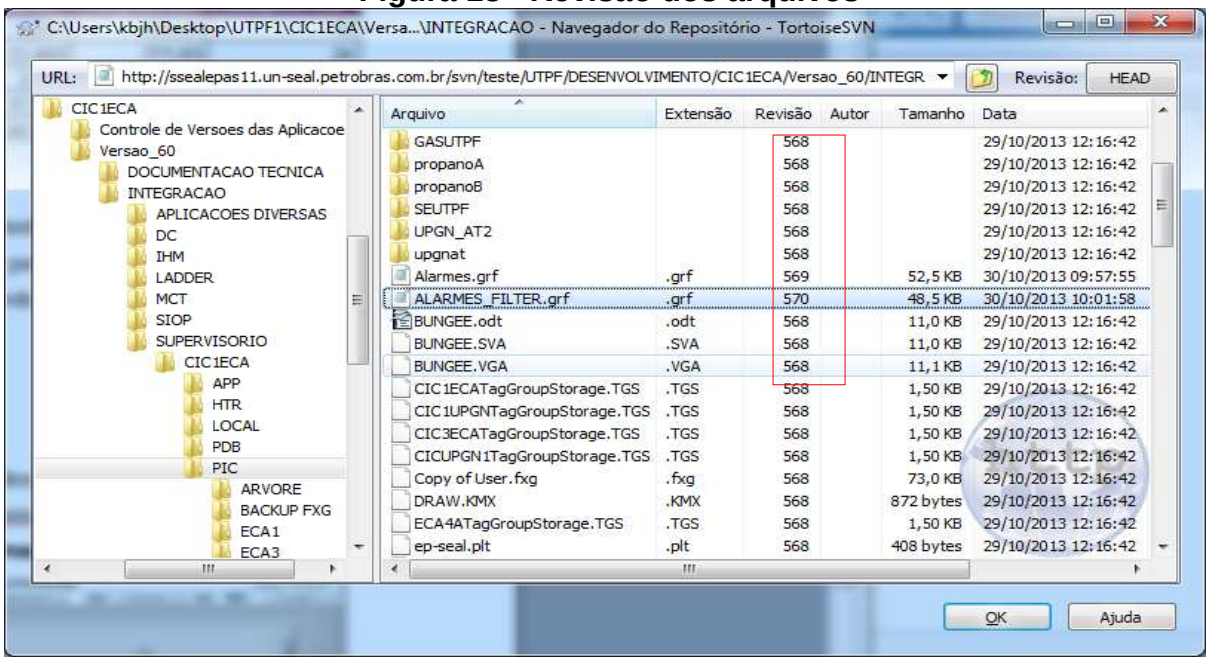
Computadores	N°DIRETÓRIOS	Total de arquivos	Carga em MB	Tempo em min.
Estação 1	3	9.557	468,41	14
Estação 2	5	13.466	771,85	24
Estação 3	2	3.569	355,35	11
Estação 4	7	17.978	854,65	27
TOTAL	17	44.570	2450,26	76

Fonte: Autor.

Após o resultado do teste de carga, se fez necessário a realização de testes do subversion quanto à integridade dos dados. O teste consistiu em modificar o conteúdo de 456 arquivos, na cópia de trabalho, de telas de supervisório e enviado

as alterações ao servidor com o comando submeter, a figura 18 mostra o momento atual do repositório e indica o numero da revisão e a data que os arquivos foram alterados pela ultima vez e também seus atributos (nome, extensão, autor, tamanho).

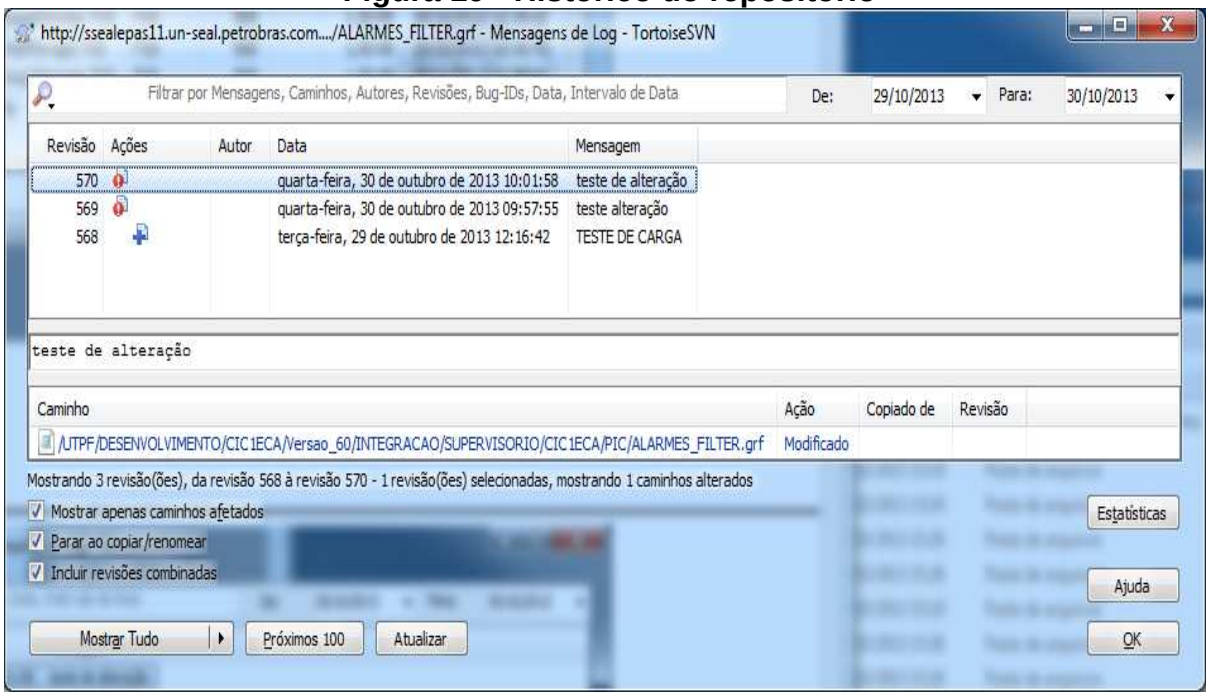
Figura 18 - Revisão dos arquivos



Fonte: Petrobrás

O histórico de revisões de um arquivo no repositório está demonstrado na figura 19.

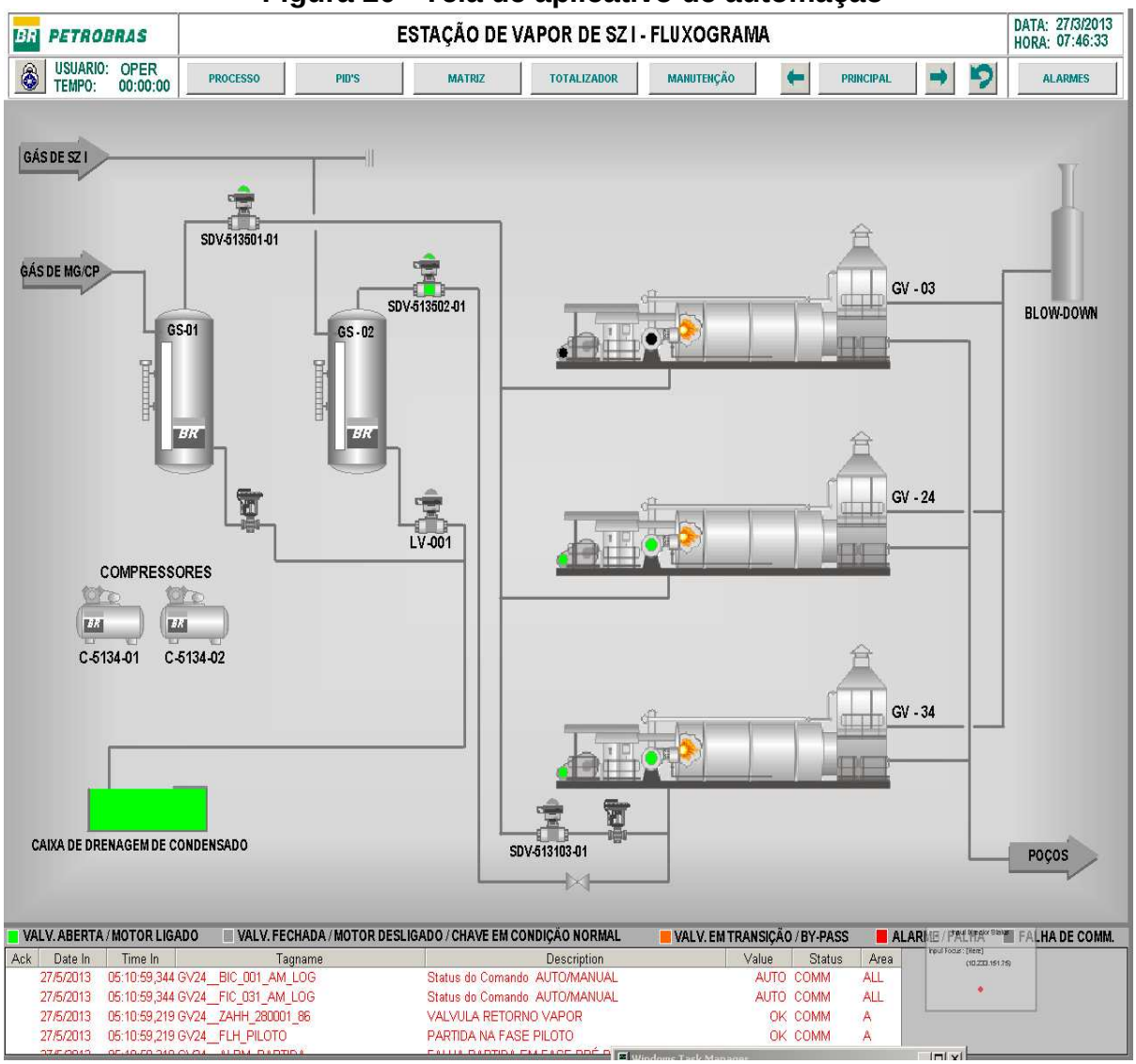
Figura 19 - Histórico do repositório



Fonte: Petrobrás.

Foram verificadas a integridade e confiabilidade dos arquivos nas versões alteradas com os programas específicos de automação, um exemplo de tela verificada é mostrado na figura 20.

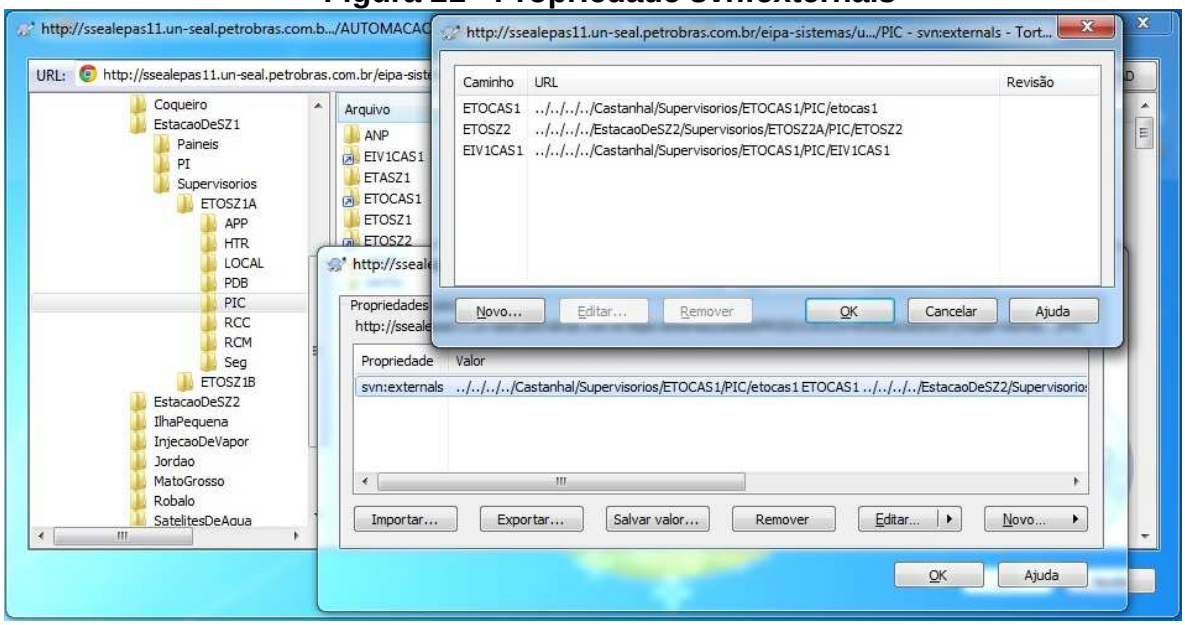
Figura 20 - Tela de aplicativo de automação



Fonte: Petrobrás

Para verificar a eficácia da propriedade *svn:externals* foram realizados testes de edição dos relacionamentos entre as aplicações. Esta propriedade tem como finalidade a replicação de um conjunto de arquivos de telas de aplicativos de supervisório servidor SCADA (Sistemas de Supervisão e Aquisição de Dados) para várias aplicações clientes VISTAS. Este procedimento sempre foi realizado forma manual, copiando-se as telas em cada uma das aplicações e a cada alteração faz-se novas copias, o objetivo do teste é manter apenas um conjunto de arquivos que ficará disponível, no repositório, para as outras aplicações através de *links simbólicos* editados na propriedade svn:externals, demonstrado na figura 21.

Figura 21 - Propriedade svn:externals



Fonte: Petrobrás

4.4 Planos de Ação

Esta etapa do projeto define as ações necessárias para implementar a ferramenta de controle de versão. Foram desenvolvidos dois planos de ação, para este objetivo. O primeiro deles define as tarefas a serem cumpridas na implantação, e o segundo define o novo plano de gestão de configuração com a inclusão da ferramenta no processo.

4.4.1 Plano de implantação da ferramenta de controle de versão

Define a seqüência e as tarefas a serem executadas para implantar a ferramenta no ambiente de automação:

1. Solicitar a criação do repositório
2. Cria e disponibilizar repositório
3. Treinamento dos usuários
4. Alterar arquitetura de rede da automação
5. Criar no repositório a estrutura de diretórios
6. Carregar o repositório com os itens de configuração
7. Liberar o acesso ao repositório
8. Instalar ferramenta cliente nas máquinas de campo

9. Editar relacionamento entre as aplicações
10. Liberar o acesso ao repositório para as equipes de manutenção

4.4.2 Plano de gestão da configuração

Foi desenvolvido pela equipe de automação para servir de guia na implementação das mudanças no processo de gerenciamento de configuração, com a implementação da ferramenta de controle de versão.

4.4.2.1 objetivo do documento

A aplicação deste plano garante a integridade dos aplicativos de automação da empresa, permitindo o acompanhamento destes itens durante todo o seu ciclo de vida, e preservando o histórico de evolução dos sistemas de automação da empresa.

O Plano de Gestão de Configuração auxilia os profissionais envolvidos a:

• controlar as mudanças em itens de configuração;

• rastrear modificações nos itens de configuração ao longo do tempo.

Apesar de serem englobadas pela Gerência de Configuração, as seguintes informações não serão tratadas neste plano, e deverão ser estabelecidas em momento futuro:

• itens de configuração de hardware;

• políticas de controle de mudanças.

4.4.2.2 aplicação

O Plano de Gestão de Configuração tem aplicação direta nas atividades dos profissionais de desenvolvimento, fiscalização e manutenção dos sistemas de automação.

4.4.2.3 ferramentas utilizadas

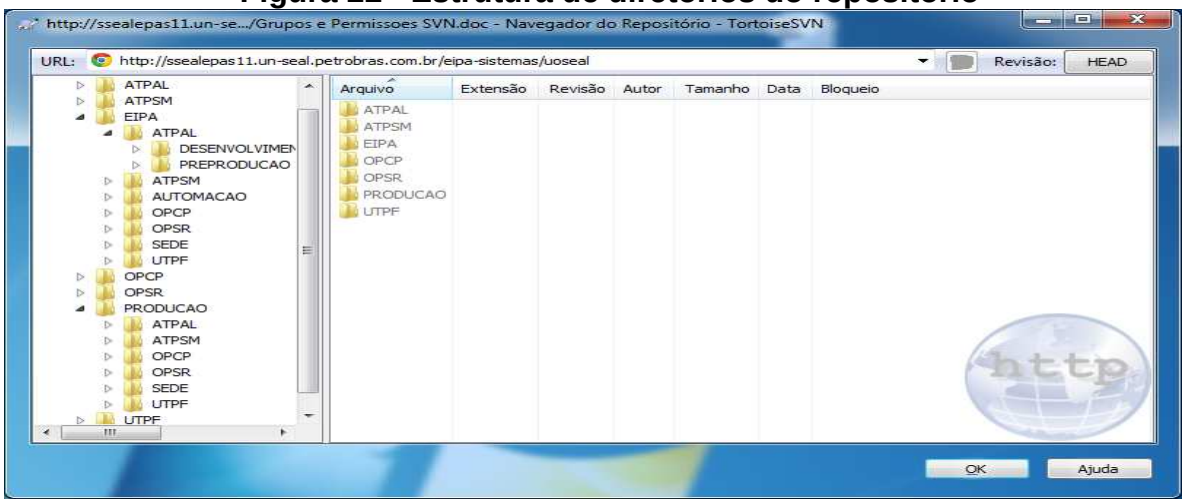
As ferramentas para controle de versão utilizadas na automação serão o Subversion e o TortoiseSVN.

4.4.2.4 repositório do sistema

O sistema de automação possuirá um repositório, contendo ambientes de desenvolvimento, pré-produção e produção.

A estrutura de diretórios, apresentada na figura 22, contempla as necessidades de praticamente todos os usuários dos sistemas de automação. Entretanto, caso necessário, deverá ser adaptada a fim de cobrir necessidades específicas não previstas.

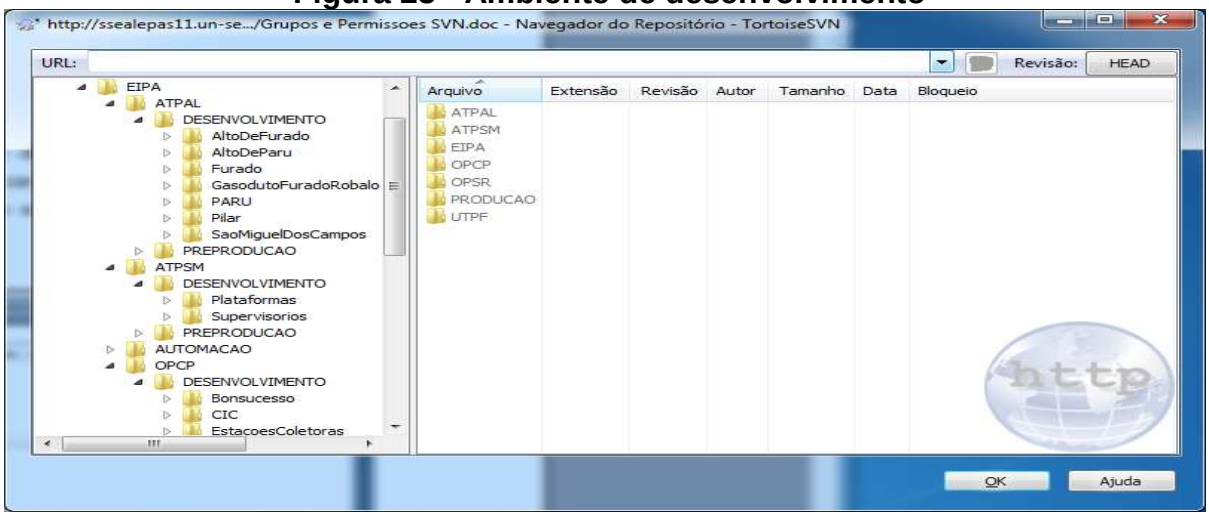
Figura 22 - Estrutura de diretórios do repositório



Fonte: Petrobrás.

O ambiente de desenvolvimento é local de trabalho da equipe de desenvolvimento, onde ocorre a elaboração e atualização dos itens de configuração. Toda a equipe de desenvolvimento e fiscalização tem acesso aos itens de configuração deste ambiente, mostrados na figura 23.

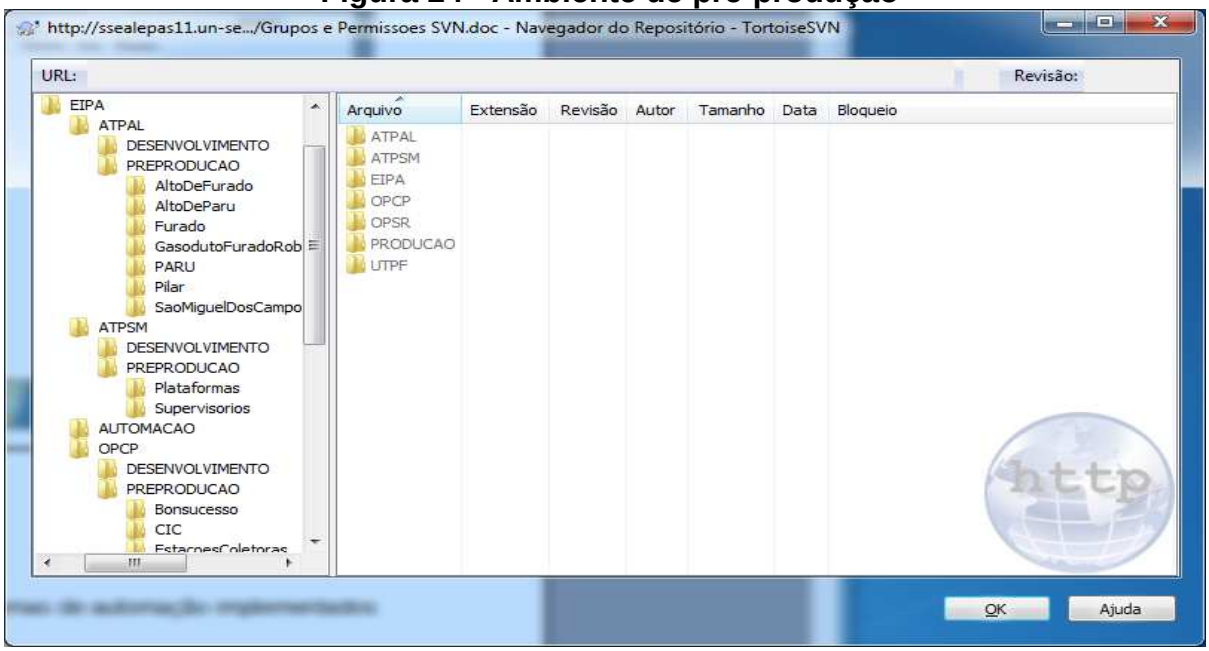
Figura 23 - Ambiente de desenvolvimento



Fonte: Petrobrás.

O segundo ambiente é o de pré-produção. Este ambiente contém versões dos sistemas de automação testadas e homologadas e que aguardam implementação no ambiente de produção (Campo) que esta demonstrado figura 24.

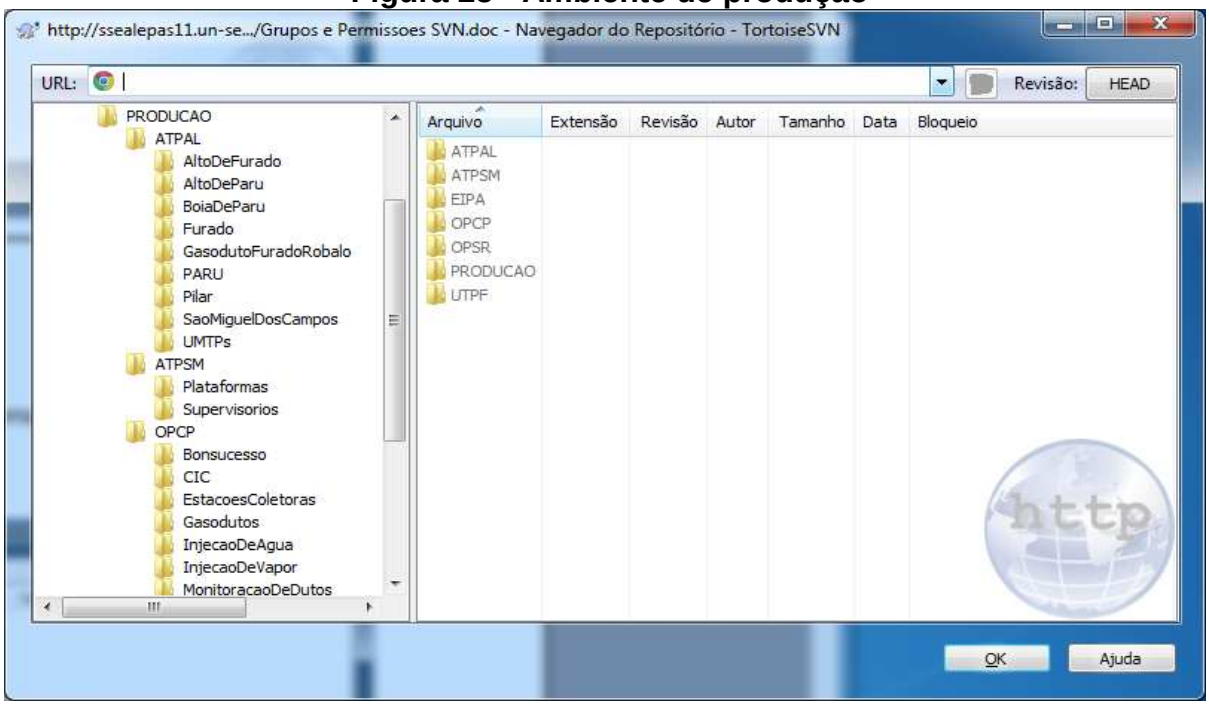
Figura 24 - Ambiente de pré-produção



Fonte: Petrobrás.

O terceiro ambiente é o de produção. Contém apenas versões dos sistemas de automação implementados em campo, demonstrado na figura 25.

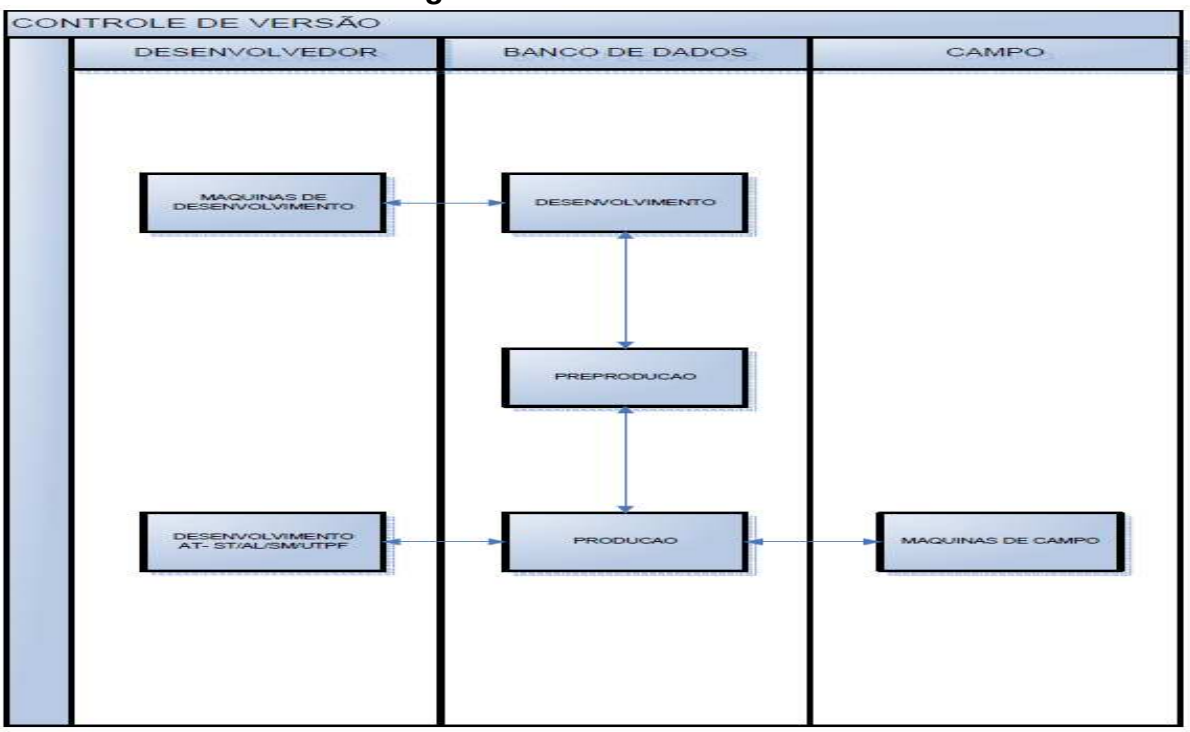
Figura 25 - Ambiente de produção



Fonte: Petrobrás

O fluxo de dados para controle de versão está representado na figura 26.

Figura 26 - Fluxo de dados



Fonte: autor

4.4.2.5 itens de configuração

Os itens de configuração sob gerência de configuração dos sistemas de automação que serão enviados ao repositório de controle de versão serão:

- Relatório de testes de plataforma;
- arquivos de configuração de redes de automação;
- scripts de Atualização;
- Programas de CLP;
- Aplicativos de Supervisório;
- Aplicativos de IHM;
- Aplicativos do PI (Plant Information).

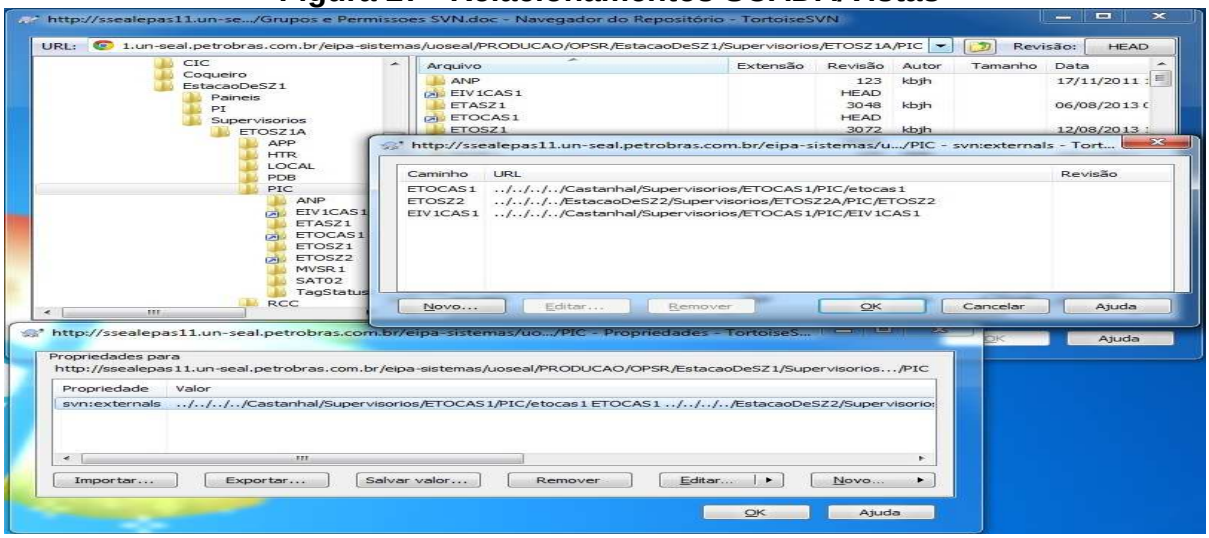
Havendo necessidade de gerência de outros itens não listados acima, devem ser definidos pelo Gerente de Configuração e inseridos no repositório do sistema.

4.4.2.6 relacionamentos entre aplicações

Os relacionamentos entre as aplicações SCADA e Vistas serão editados

na pasta PIC das aplicações Vistas, utilizando-se a propriedade svn:externals da ferramenta de controle de versão, demonstrado na figura 27.

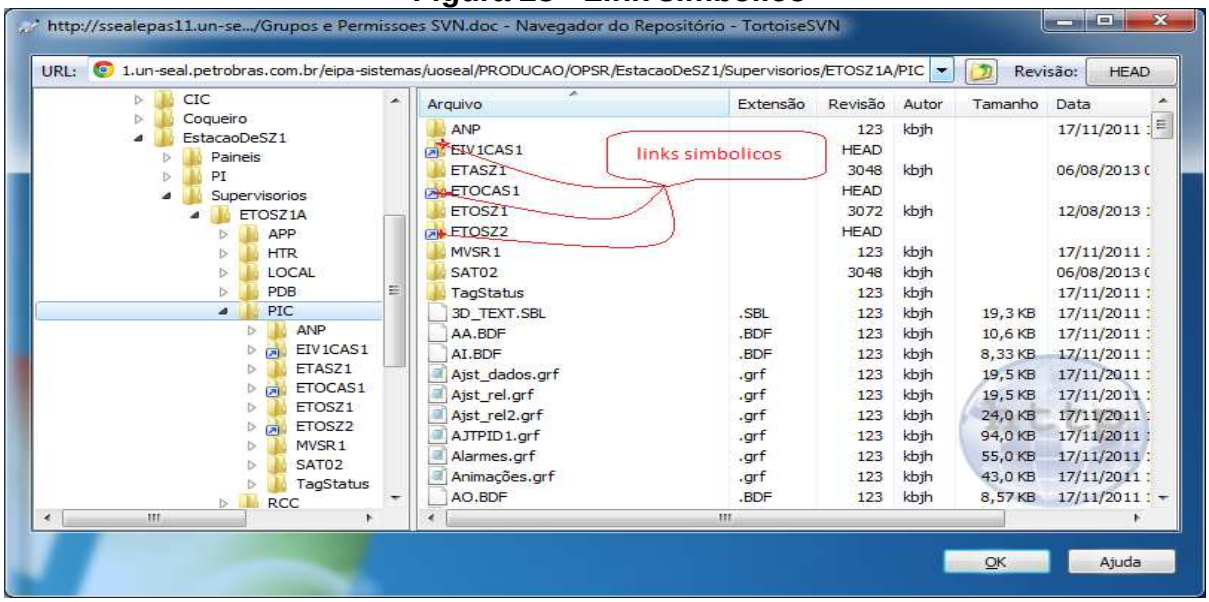
Figura 27 - Relacionamentos SCADA/Vistas



Fonte: Autor

Deste modo a replicação da pasta PIC dos SCADAs para as pastas PIC dos Vistas passa a ser apenas um link simbólico, e quais quer modificação realizada na pasta PIC do SCADA aparecerá nas pastas PIC dos VISTAS, demonstrado na Figura 28.

Figura 28 - Link simbólico



Fonte: Autor

Com este procedimento, será possível ter dentro do repositório apenas arquivos na pasta PIC dos SCADAs, eliminando assim as varias cópias das aplicações Vistas, o objetivo é reduzir a zero o índice de aplicações vistas desatualizadas por copias não realizadas.

4.4.2.7 papéis e responsabilidades

Neste item serão abordados aspectos referentes aos usuários dos sistemas de controle de aplicativos de automação.

4.4.2.7.1 gerente de configuração

O gerente do contrato de automação deve indicar um profissional da equipe para exercer o papel de gerente de configuração. Este profissional deve apresentar o seguinte perfil:

• experiência prática nas ferramentas de gerência de configuração utilizadas (controle de versão, controle de mudanças, etc.);

• conhecimento das políticas, diretrizes e procedimentos de desenvolvimento dos aplicativos de automação.

O Gerente de Configuração é responsável por:

• solicitar a criação de repositórios;

• estruturar o ambiente de desenvolvimento no repositório;

• identificar e controlar os itens de configuração;

• disponibilizar treinamento dos usuários;

• apoiar a equipe de desenvolvimento no uso das ferramentas de gerência de configuração;

• elaborar padrão de registro das alterações implementadas em cada versão disponibilizada do sistema.

4.4.2.7.2 equipe de desenvolvimento

A equipe de desenvolvimento mantém os itens de configuração no ambiente de desenvolvimento do repositório do sistema através de suas atividades diárias. Suas atividades incluem:

• substituição das versões de produção dos sistemas por novas versões disponibilizadas pelos Fiscais de contrato;

• descrever todos os passos para geração, instalação e inicialização segura do sistema;

• descrever todos os procedimentos para manter a segurança ao substituir

uma versão no ambiente de Produção.

• garantir que o sistema recebido pela produção corresponda precisamente à versão correta do sistema;

• evitar ou detectar qualquer erro da versão atual do sistema;

• evitar divulgação não autorizada das versões do sistema;

4.4.2.7.3 fiscais de contrato

Os fiscais de contrato são os responsáveis por autorizar a atualização de uma versão de produção em campo e auditar as versões desenvolvidas sob sua responsabilidade.

4.4.2.7.4 equipes de manutenção

As equipes de manutenção dos ativos de produção ficam responsáveis por manter os aplicativos de automação atualizados e quaisquer alterações realizadas devem ser enviadas ao repositório.

4.4.2.7.5 infra-estrutura de TI

A área de infra-estrutura é responsável pela disponibilização e guarda dos repositórios dos sistemas e dos servidores onde estão localizados.
Suas atividades incluem:

• criação de repositórios;

• execução de processos operacionais como: backup e restauração, configuração e atualização de servidores, administração de usuários e permissões.

4.4.2.8 restrições de acesso

Considerando um conjunto definido de papéis exercidos pelos membros das equipes e diferentes áreas da empresa, é possível definir permissões comuns de acesso aos diferentes diretórios do repositório.

A preocupação com a autenticação e autorização no repositório é fundamental para a segurança dos aplicativos desenvolvidos.

O quadro 5 relaciona a estrutura básica do repositório com os papeis envolvidos nas atividades deste plano, indicando o tipo de permissão/restrição necessária para cada caso.

Quadro 5 - Permissão/restrição

permissão/restrição				
	Equipes do Sistema			
Diretórios	Gerente de Configuração	Fiscalização	Desenvolvimento	Manutenção
EIPA	L-E	L-E	L-E	L
<Ativo de Produção>	L-E	L-E	L	L-E
Produção/<Ativo de Produção>	L-E	L-E	L-E	L-E
L = leitura / E = escrita				

Fonte: autor

Apenas o Gerente de Configuração tem permissão para criar uma nova ramificação para um novo projeto, entretanto toda a equipe de desenvolvimento tem permissão para manter os itens de configuração desta ramificação.

4.5 Implantar a Ferramenta de Controle de Versão no Ambiente de Automação da Empresa.

As tarefas foram realizadas de acordo com os planos de implementações, os quais descrevem as tarefas a serem executadas e o processo de gestão de configuração dos aplicativos de automação.

4.5.1 Solicitar a criação do repositório

O gerente de configuração solicitou através de email da empresa a criação do repositório de aplicativos de automação ao departamento de infra-estrutura.

4.5.2 Criar e disponibilizar repositório

O departamento de infra-estrutura de TI criou e configurou o repositório para controle de versão dos aplicativos de automação, também disponibilizou o endereço do servidor para acesso ao repositório.

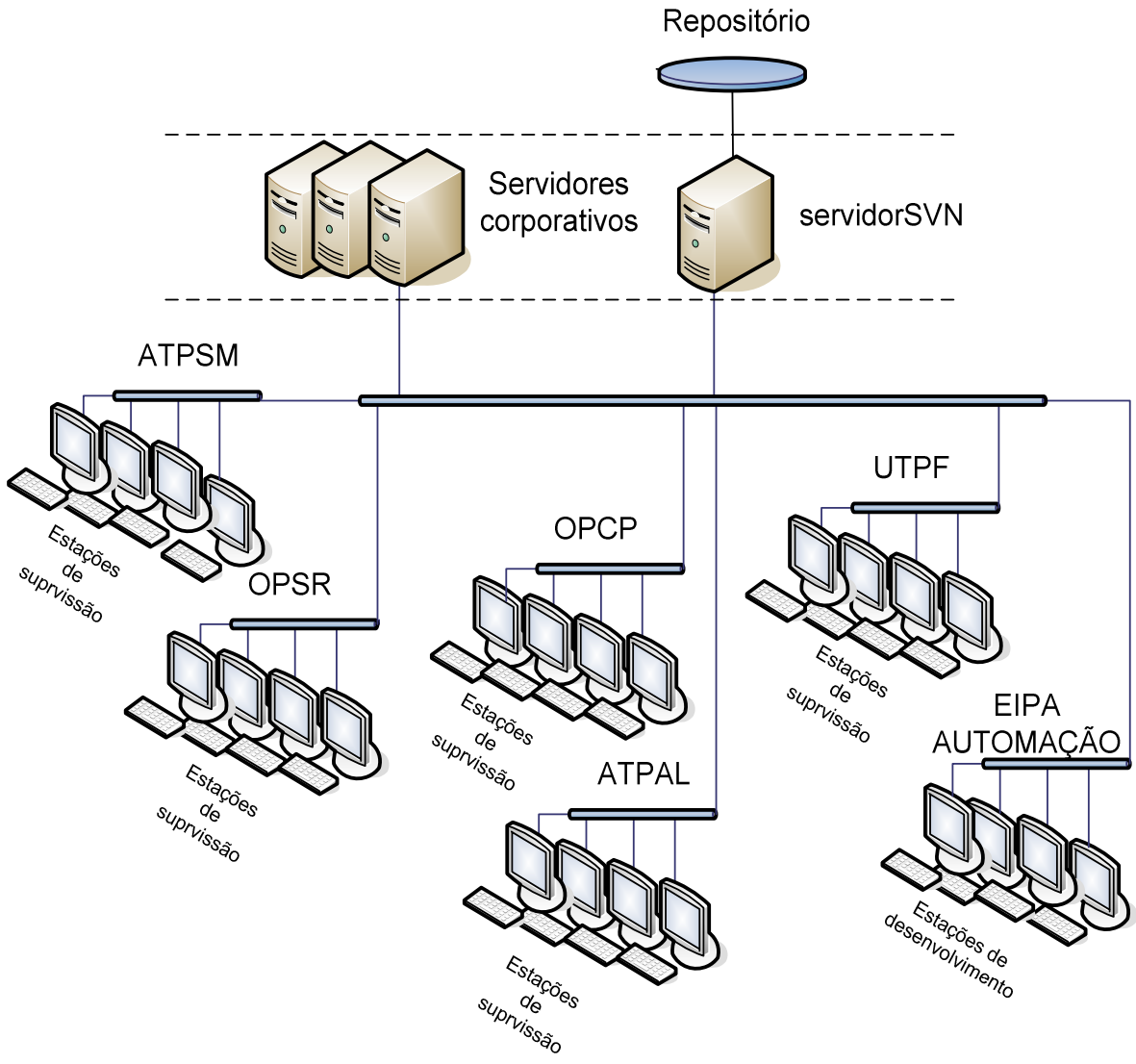
4.5.3 Treinamento dos usuários

A equipe de infra-estrutura preparou um treinamento de utilização das ferramentas implementadas e, foi providenciada pelo gerente de configuração junto à área de recursos humanos a disponibilização dos recursos necessários para a realização do treinamento dos usuários.

4.5.4 Alterar arquitetura de rede da automação

Foi adicionado na arquitetura de automação um repositório e o servidorSVN como se vê na figura 29.

Figura 29 - Rede de Automação

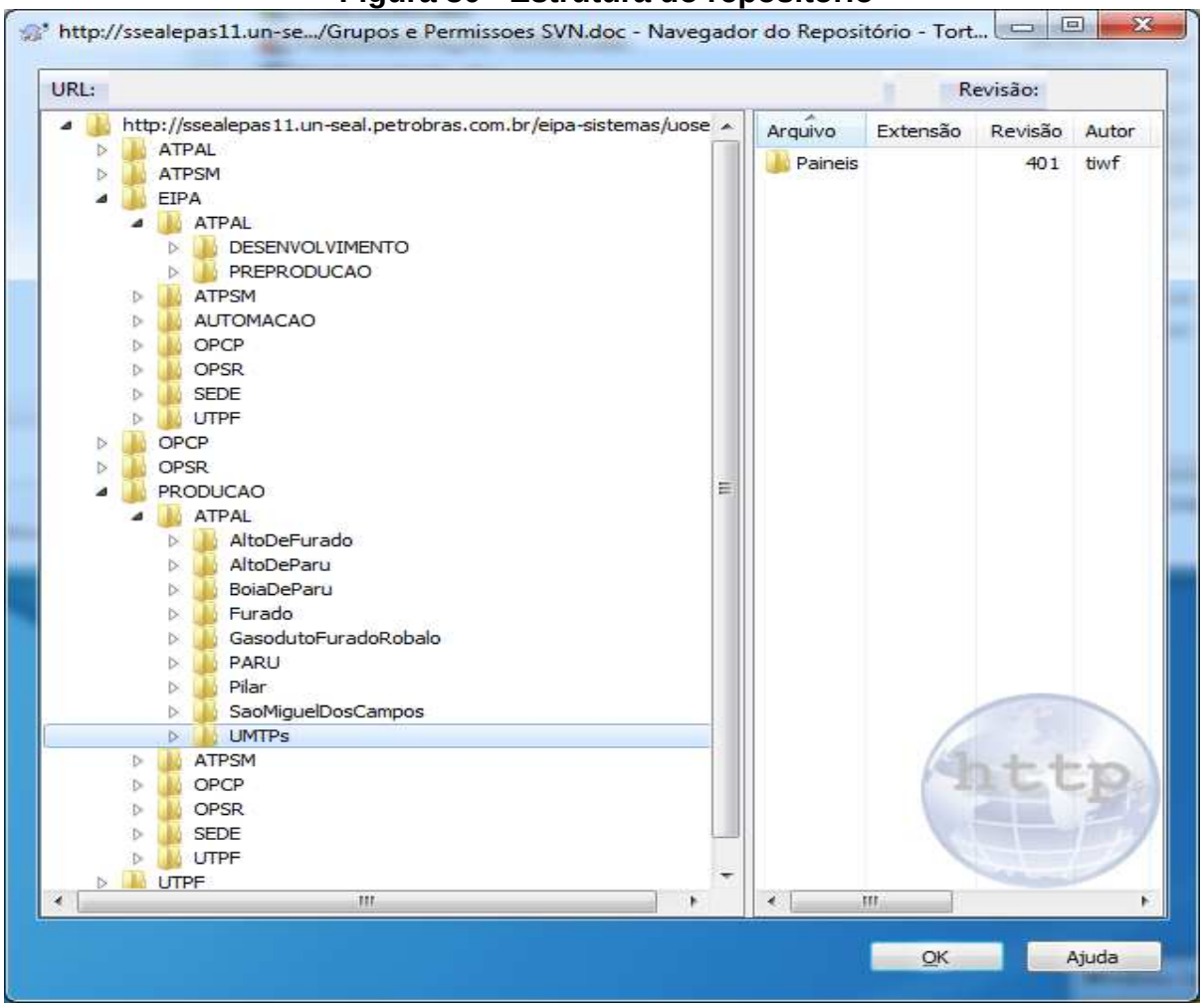


Fonte: autor

4.5.5 Criar no repositório a estrutura de diretórios

Foi criado no repositório pelo gerente de configuração, através da ferramenta TortoiseSVN a estrutura de diretórios definida pela equipe de automação no item 5.6.4, representado na figura 30.

Figura 30 - Estrutura do repositório

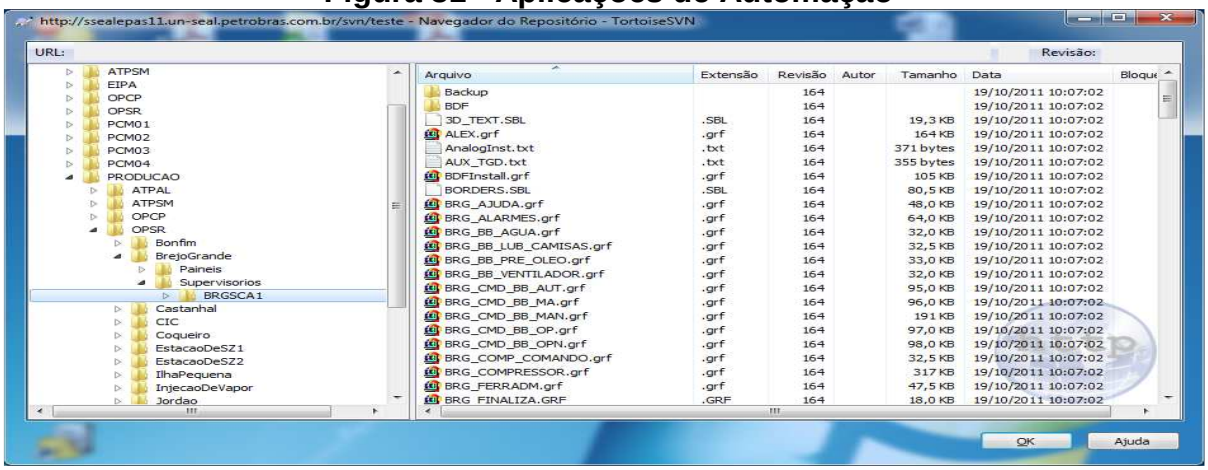


Fonte: autor

4.5.6 Carregar o repositório com os itens de configuração

Foram selecionadas a ultima versão dos backups de cada aplicativo de automação encontrados na área de armazenamento dos servidores da empresa. A migração das aplicações para o repositório foi realizada pelos fiscais de contratos de acordo com sua área de responsabilidade, num total de 1.364.844 arquivos, dentre eles estão: 167 aplicações de supervisão, 390 programas de CLP, 230 aplicações de IHM e outras aplicações. O modelo de aplicação foi demonstrado na figura 31.

Figura 31 - Aplicações de Automação

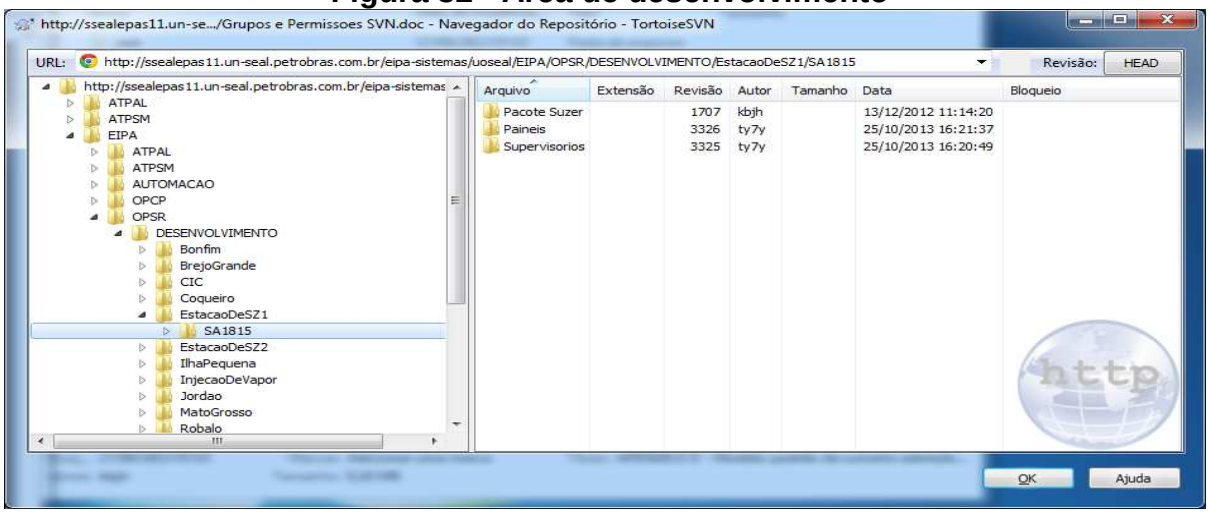


Fonte Autor.

4.5.7 Liberar o acesso ao repositório

Estruturado no repositório a área de desenvolvimento com os projetos em andamento, conforme mostrado na figura 32. E também nessa etapa foi solicitada à área de infra-estrutura a liberação do acesso para a equipe de desenvolvimento.

Figura 32 - Área de desenvolvimento



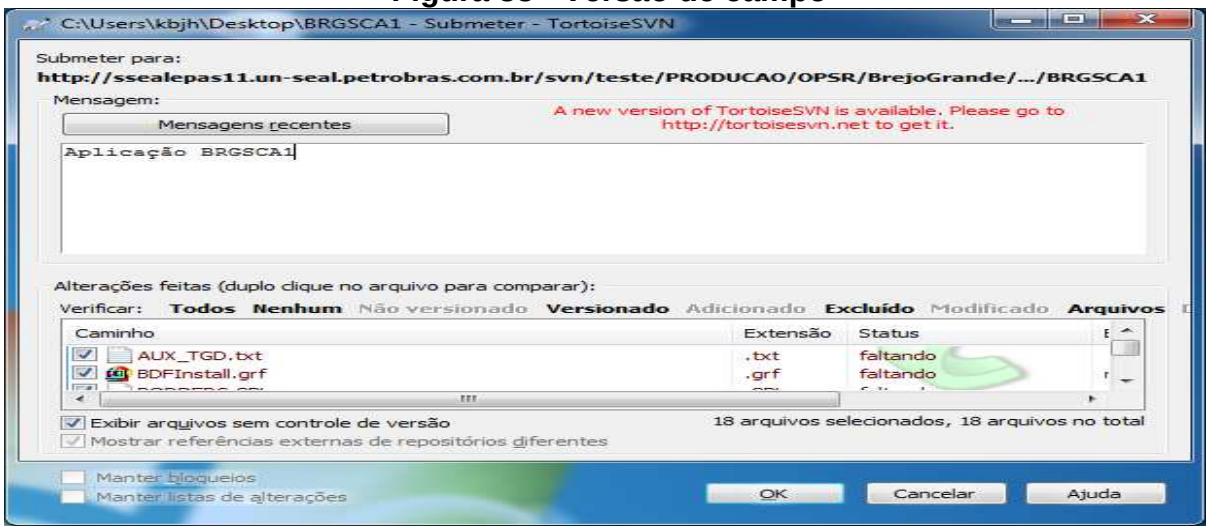
Fonte Autor.

4.5.8 Instalar ferramenta cliente TortoiseSVN nas máquinas de campo

A equipe de desenvolvimento instalou a ferramenta cliente TortoiseSVN nas 167 máquinas de supervisório do campo localizadas nos 5 ativos de produção da empresas (ATPST, ATPSR, ATPSM, ATPAL, UTPF), enviado ao repositório a versão de campo que estavam nas máquinas demonstrado na figura 33 e também

foi instalado a ferramenta cliente TortoiseSVN nas maquinas de desenvolvimento da EIPA/automação.

Figura 33 - Versão de campo

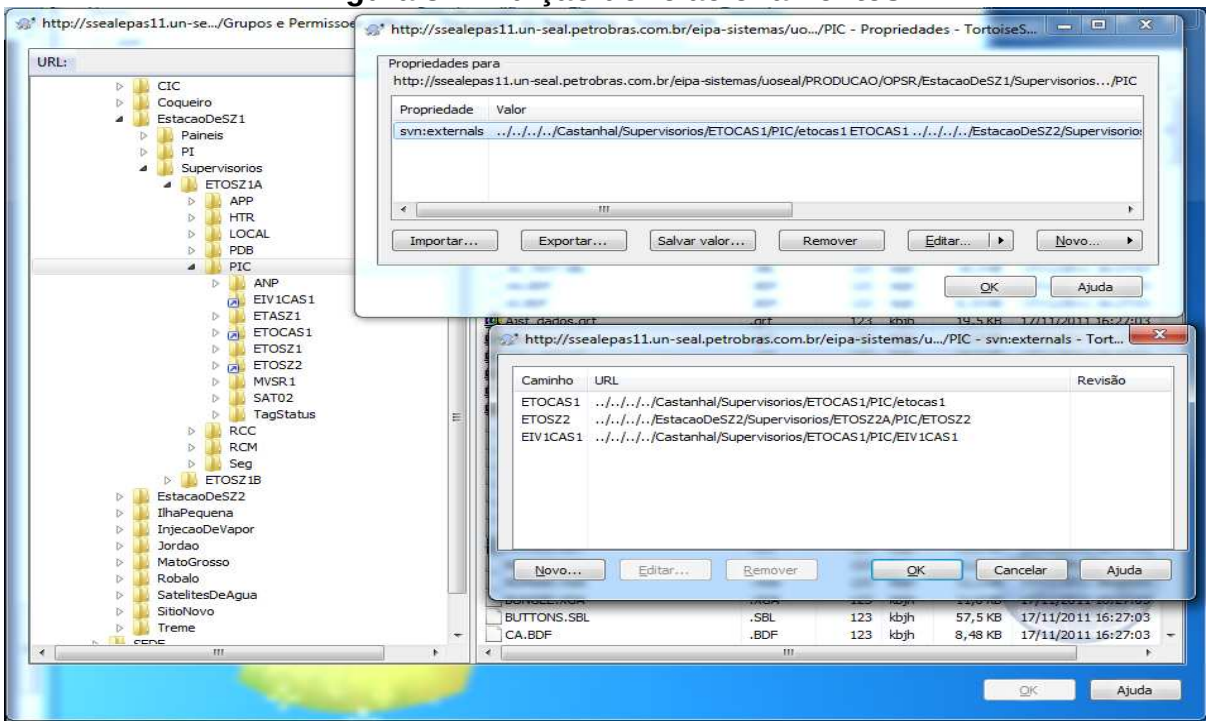


Fonte: Autor.

4.5.9 Editar relacionamento entre as aplicações

O relacionamento entre as aplicações foi implementado pela equipe de desenvolvimento em dois ativos de produção OPCP e OPSR, num total de 45 aplicações como se verifica na figura 34.

Figura 34 - Edição de relacionamentos



Fonte Autor.

4.5.10 Liberar o acesso ao repositório para as equipes de manutenção

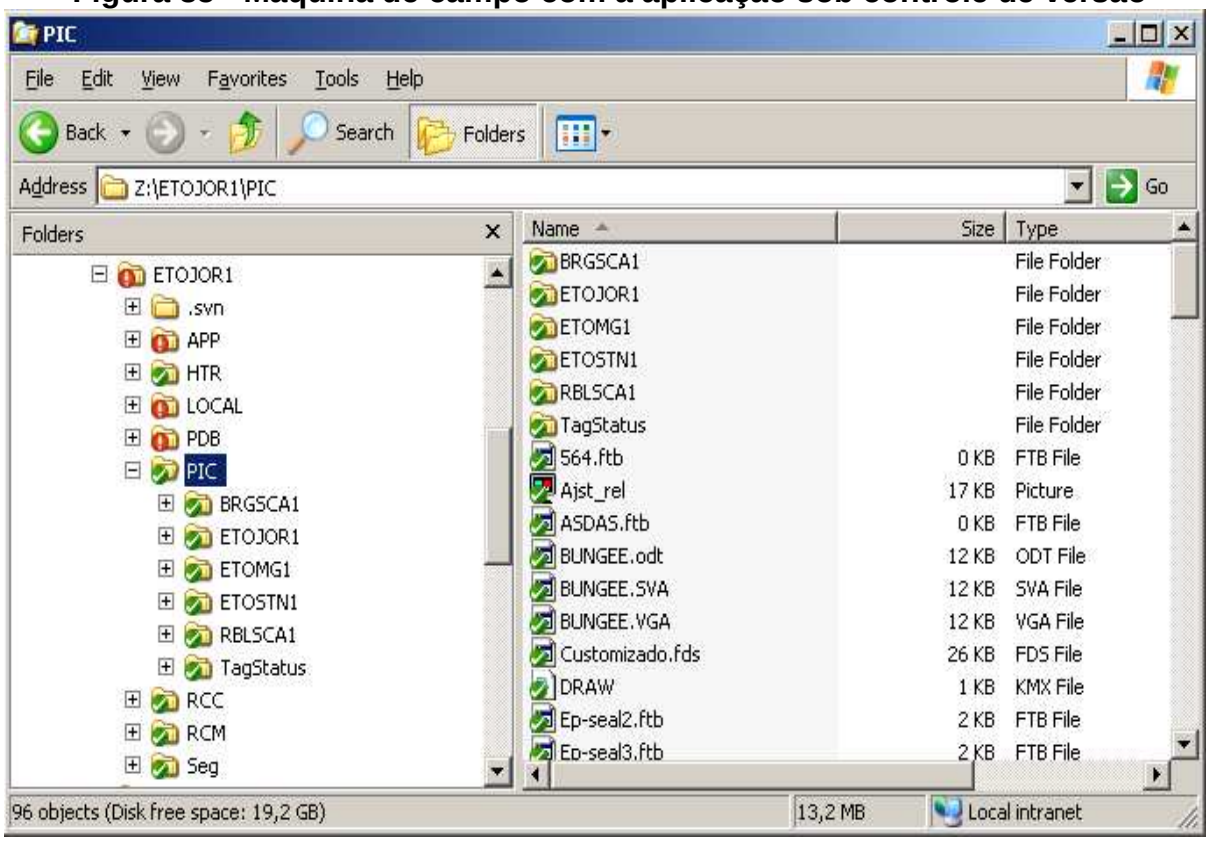
As equipes de manutenção dos ativos de produção receberam treinamento nas ferramentas e foi liberado o acesso ao repositório, treinamento e a liberação do acesso ao repositório para as equipes de manutenção dos ativos de produção encerrou-se o ciclo de implementação do sistema de controle de versão dos aplicativos de automação.

4.5.11 Verificação da implementação da ferramenta de controle de versão no ambiente de automação da empresa.

Através da ferramenta cliente TortoiseSVN, foi verificado o controle realizado pelo Subversion nas alterações dos aplicativos de automação implementadas pelas equipes de desenvolvimento e manutenção.

Na figura 35 observa-se como ficou a aplicação de Supervisório na máquina de campo da estação de Jordão (ETOJOR1), com a implementação do controle de versão.

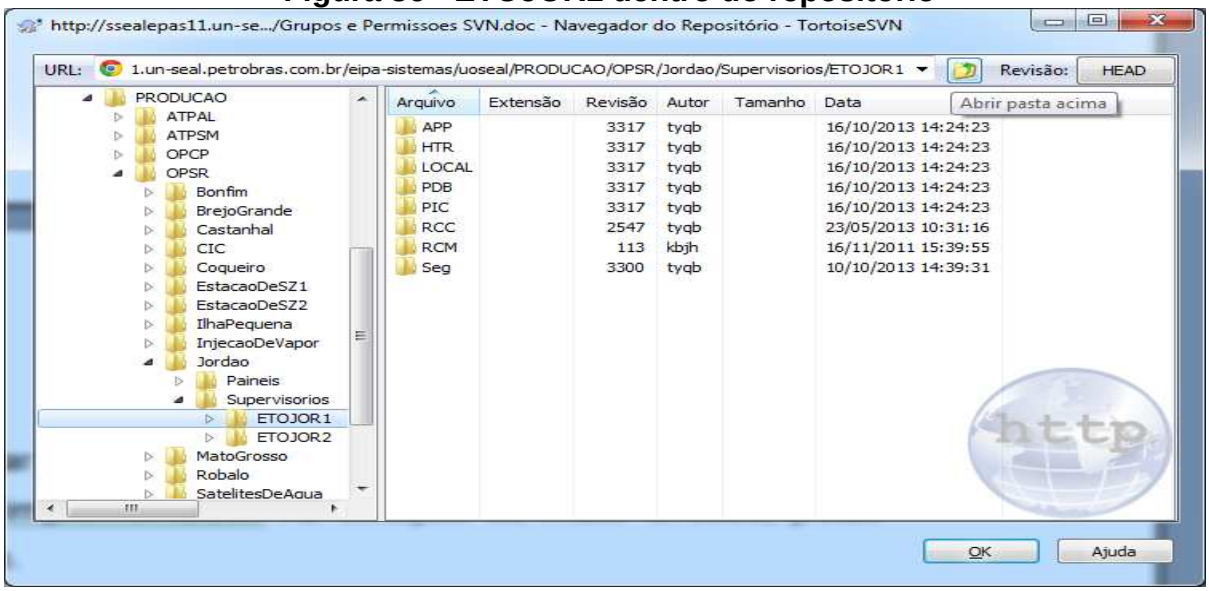
Figura 35 - Máquina de campo com a aplicação sob controle de versão



Fonte: Petrobrás.

Na figura 36 vemos a mesma aplicação de automação dentro do repositório.

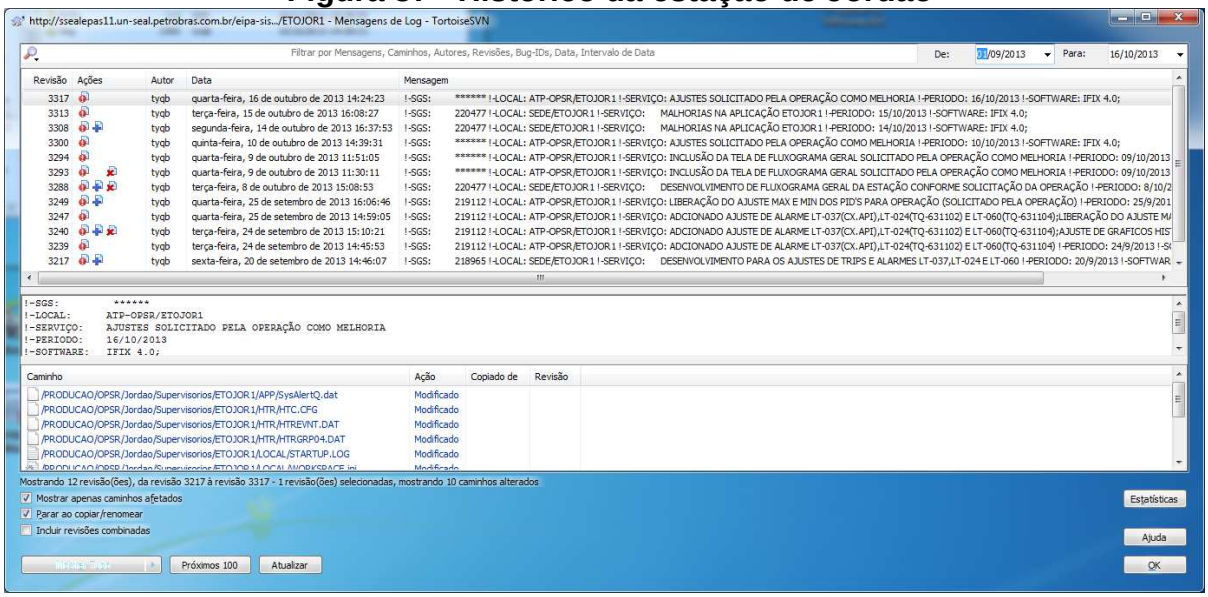
Figura 36 - ETOJOR1 dentro do repositório



Fonte Petrobrás.

Observa-se na figura 37 o histórico fornecido pela ferramenta, com base no controle das alterações realizadas na aplicação do supervisório da estação de Jordão. No histórico, tem-se informações de 12 revisões realizadas no período de 01 de setembro a 16 de outubro de 2013, indicando o autor, a data que foi realizada, a mensagem descrevendo o serviço executado e os arquivos que foram modificados em cada uma das revisões.

Figura 37 - Histórico da estação de Jordão

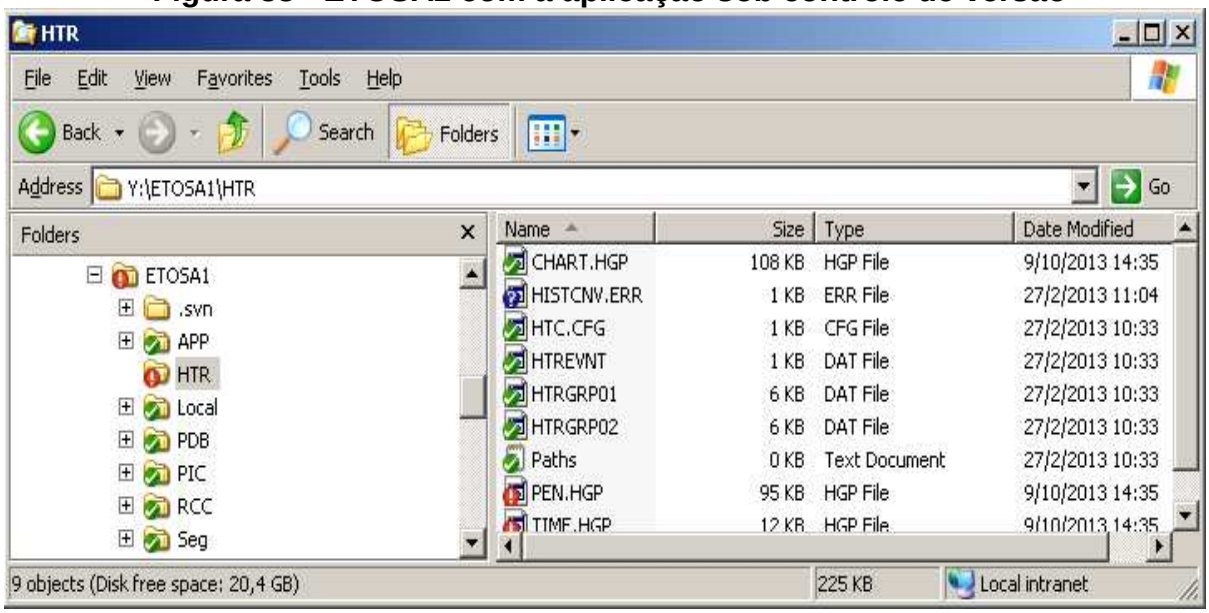


Fonte Petrobrás.

Na figura 38, verifica-se a máquina de campo da estação de Santo

Antonio (ETOSA1), com sua aplicação sob o controle de versão.

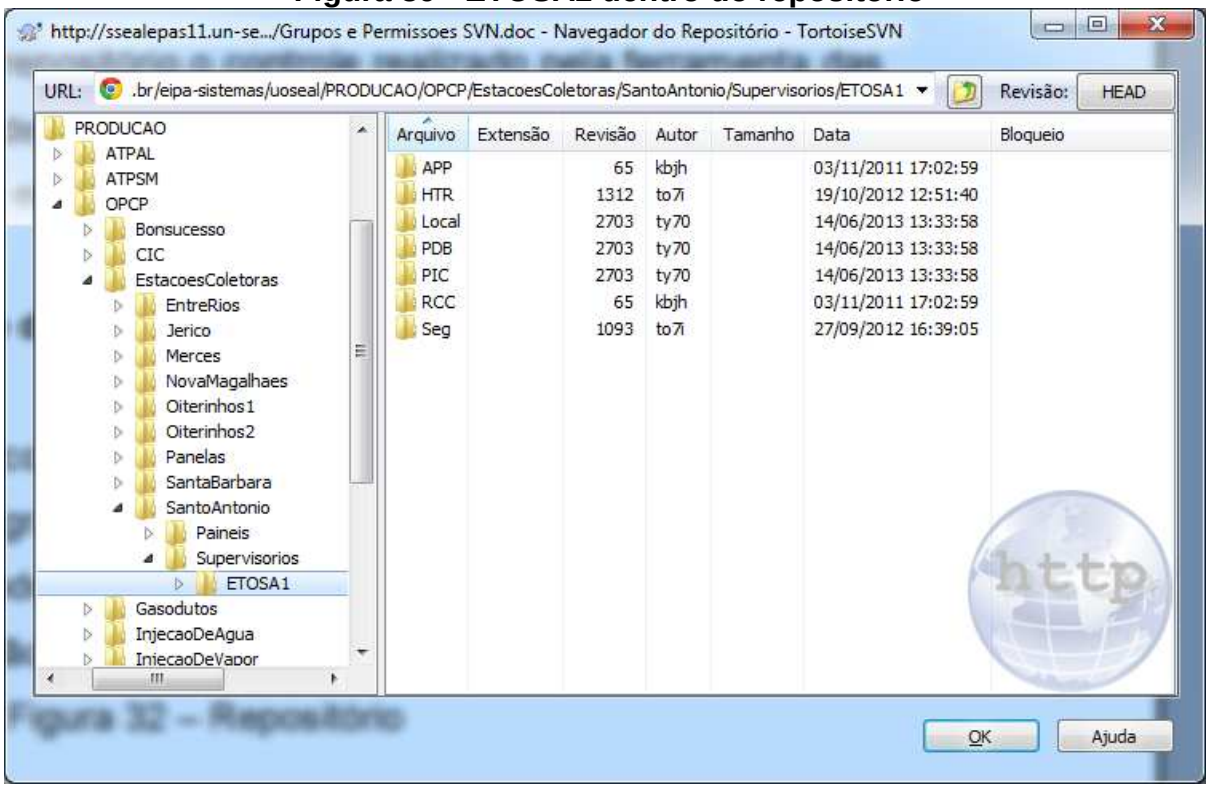
Figura 38 - ETOSA1 com a aplicação sob controle de versão



Fonte: Petrobrás.

Na figura 39, vê-se a aplicação de Santo Antonio dentro do repositório.

Figura 39 - ETOSA1 dentro do repositório

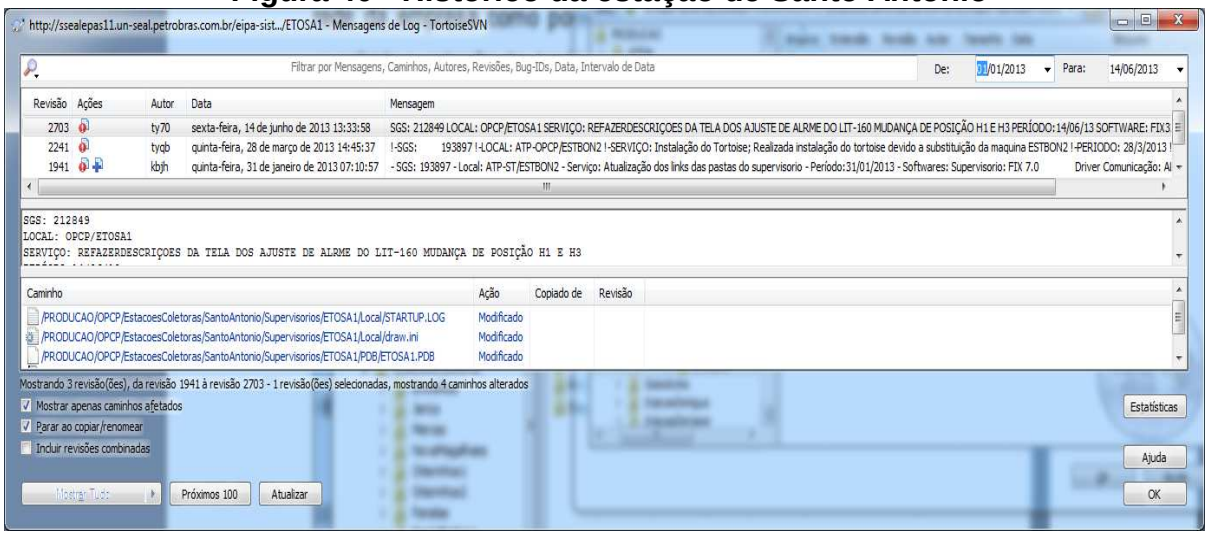


Fonte: Petrobrás.

Percebe-se, na figura 40, o histórico das alterações realizadas na aplicação do supervisório da estação de Santo Antonio. No histórico, percebe-se 3 revisões no período de 6 meses, com as principais informações de cada revisão,

autor, data de realização, mensagem e a ação realizada em cada arquivo (modificado, adicionado, excluído).

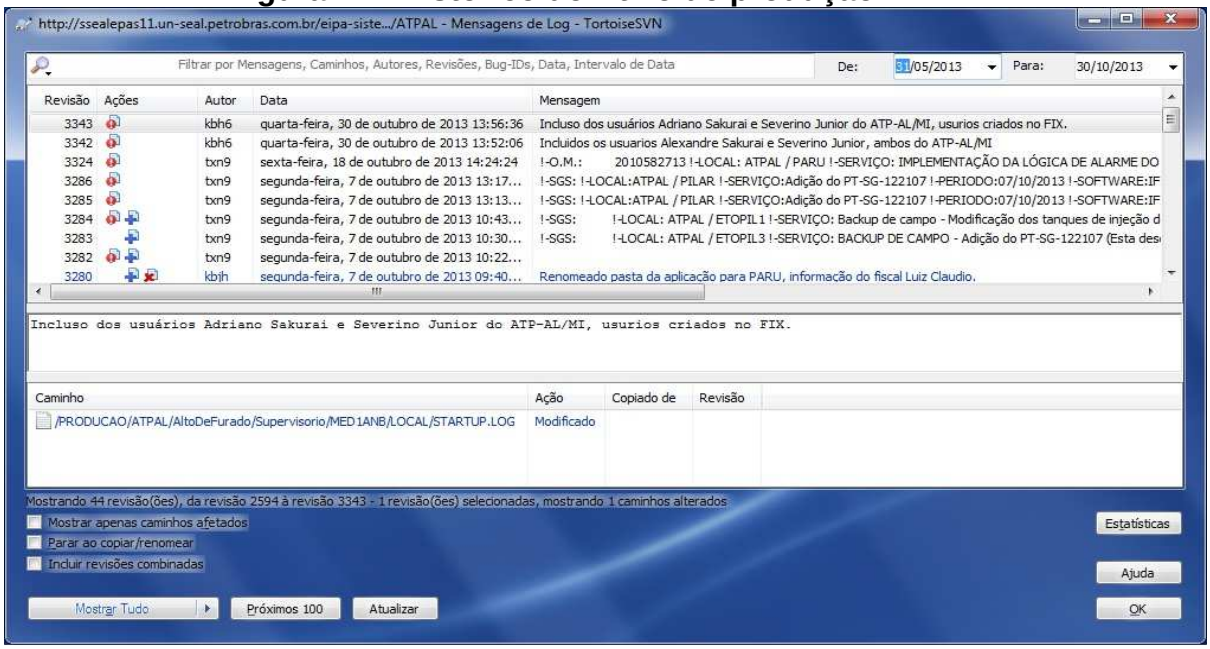
Figura 40 - Histórico da estação de Santo Antonio



Fonte: Petrobrás.

Na figura 41, tem-se o histórico de 9 das 44 Alterações realizadas no período de 31 de maio a 31 de outubro de 2013, no ativo de produção ATPAL.

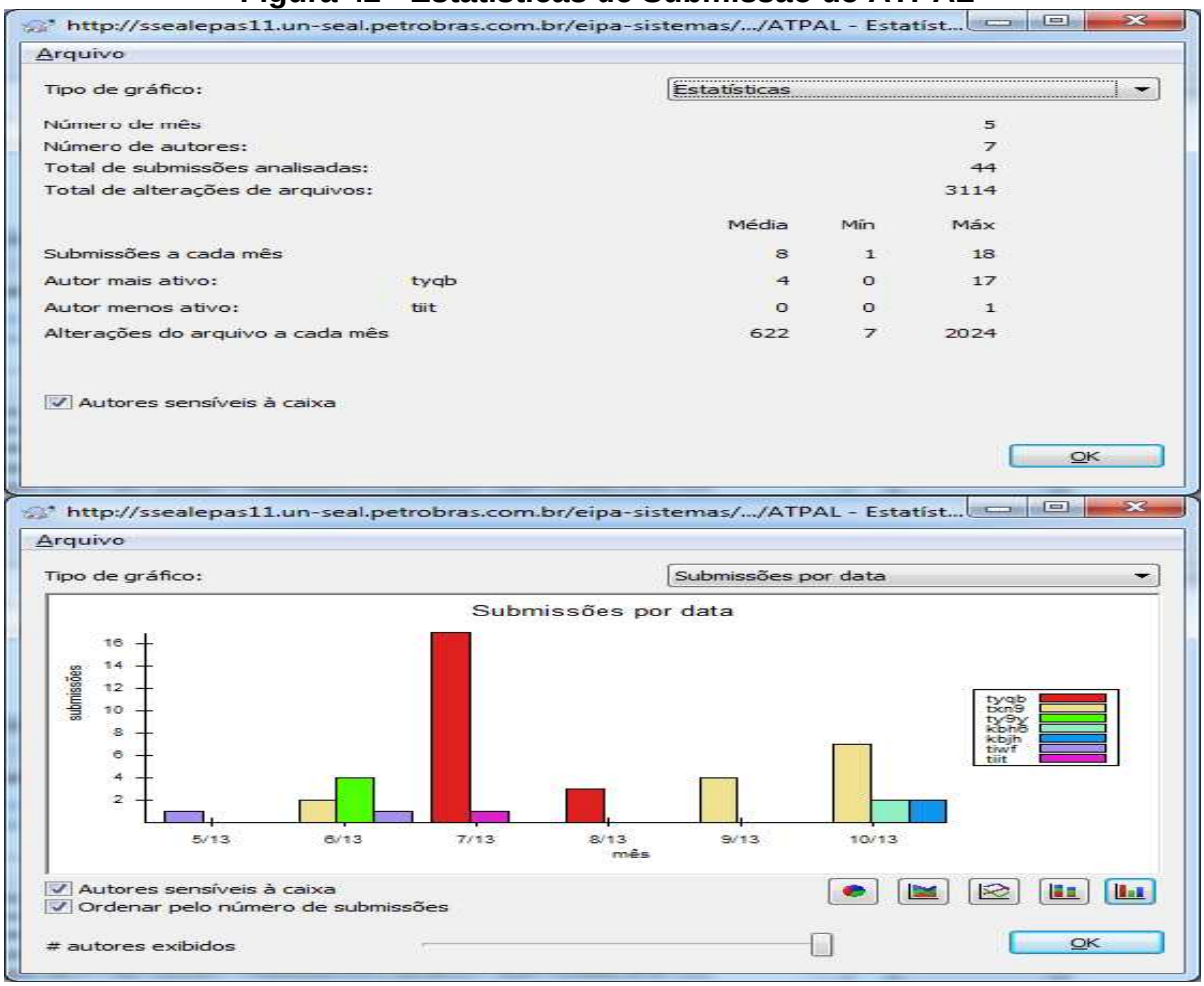
Figura 41 - Histórico do Ativo de produção ATPAL



Fonte: Petrobrás.

Para melhor analisar, o histórico o TortoiseSVN oferece ferramentas estatísticas com base neste histórico, pode-se verificar na figura 42, na qual se tem informações como, 7 autores responsáveis pelas alterações, 3114 arquivos alterados, 44 submissões analisadas, média mensal de 622 arquivos alterados, entre outros.

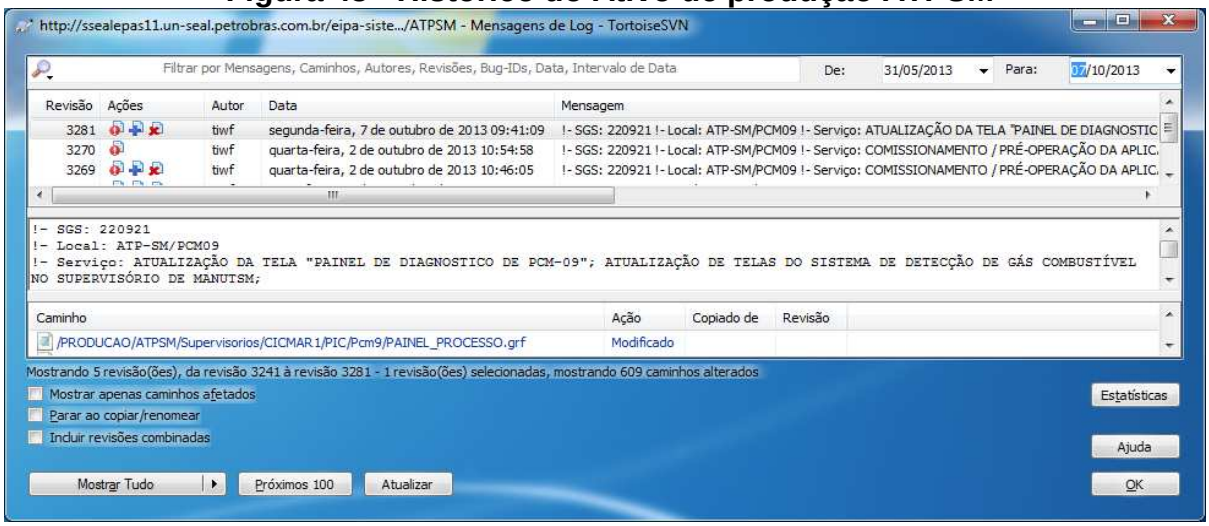
Figura 42 - Estatísticas de Submissão do ATPAL



Fonte: Petrobrás.

Na figura 43, demonstra-se o histórico das alterações realizadas no mesmo período (31 de maio a 31 de outubro de 2013), do ativo de produção ATPSM.

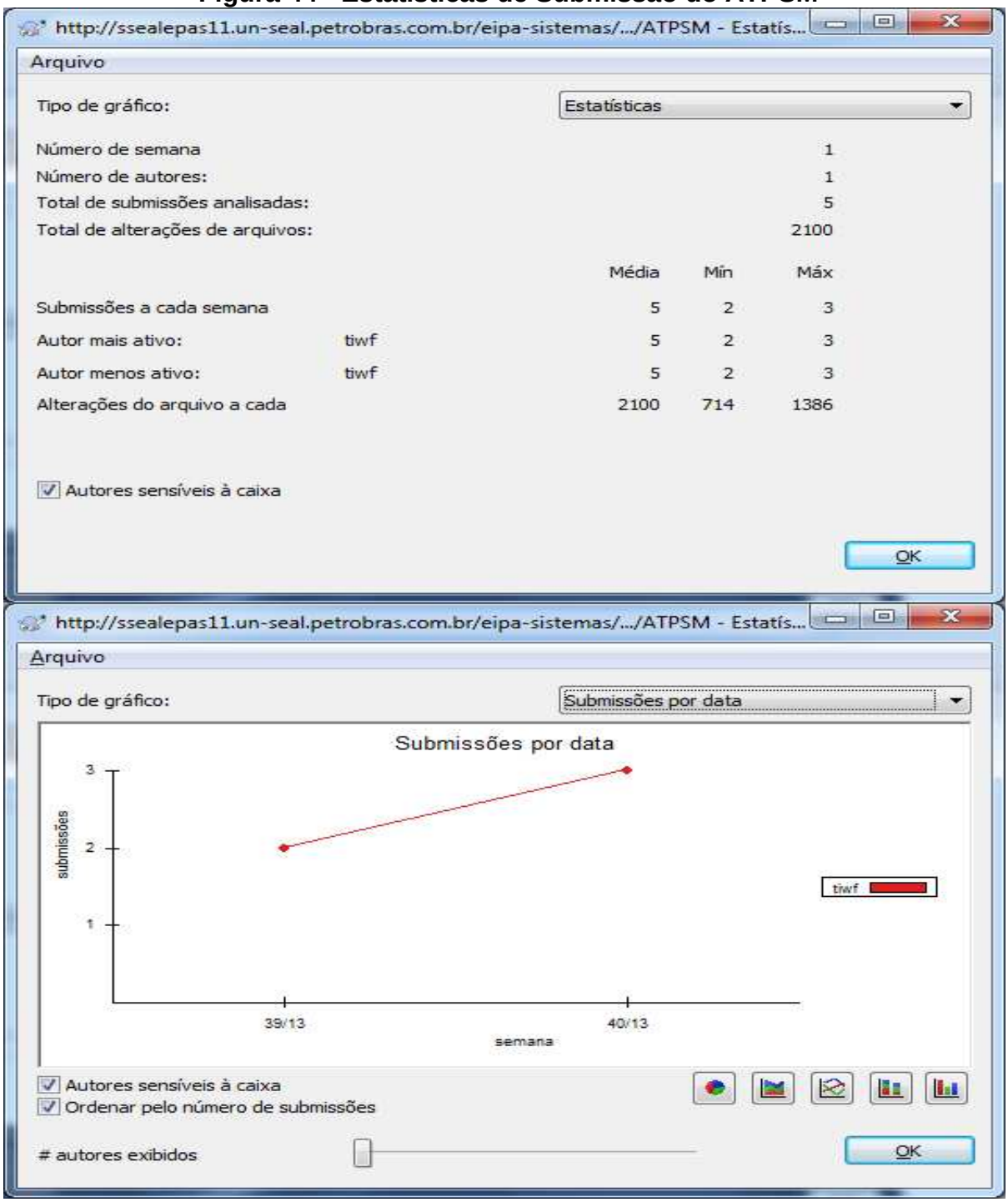
Figura 43 - Histórico do Ativo de produção ATPSM



Fonte: Petrobrás

No período mencionado foram realizadas 5 Submissões, com um total de 2100 arquivos modificados, por apenas um autor representado pela chave de acesso tiwf, conforme demonstrado na figura 44.

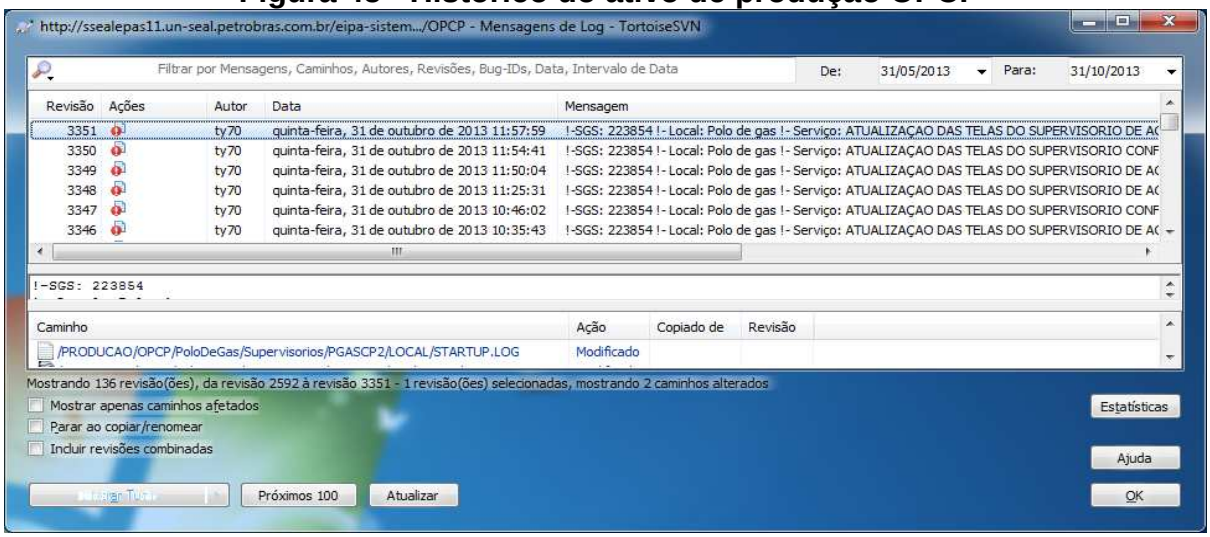
Figura 44 - Estatísticas de Submissão do ATPSM



Fonte: Petrobrás.

No ativo de produção OPCP, tem-se no mesmo período (31 de maio a 31 de outubro de 2013), um total de 136 revisões e seis delas estão demonstrados na figura 45.

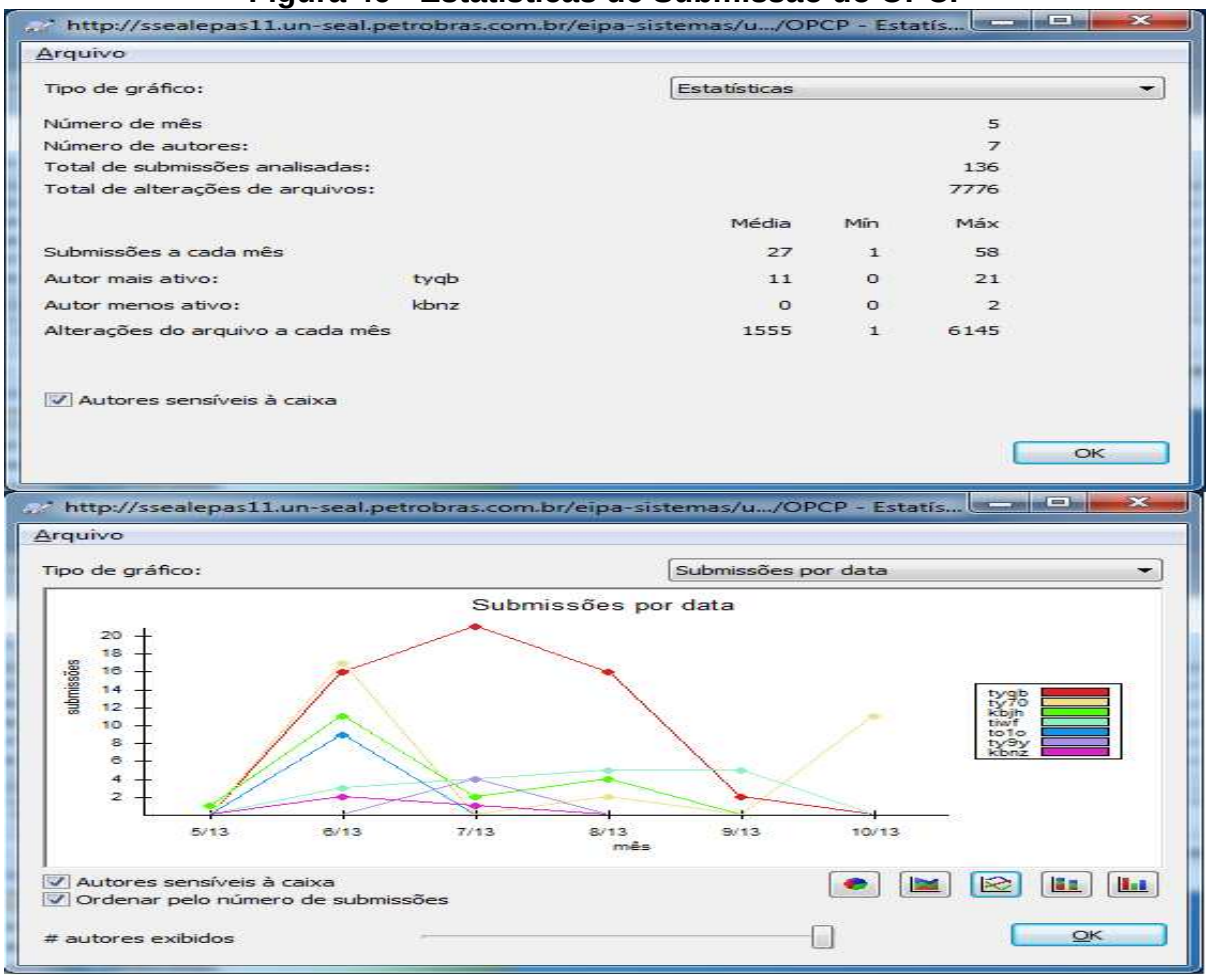
Figura 45 - Histórico do ativo de produção OPCP



Fonte: Petrobrás.

Estas 136 revisões modificaram um total de 7776 arquivos, com uma média de 27 submissões por mês, realizado por 7 autores diferentes, demonstrados na figura 46.

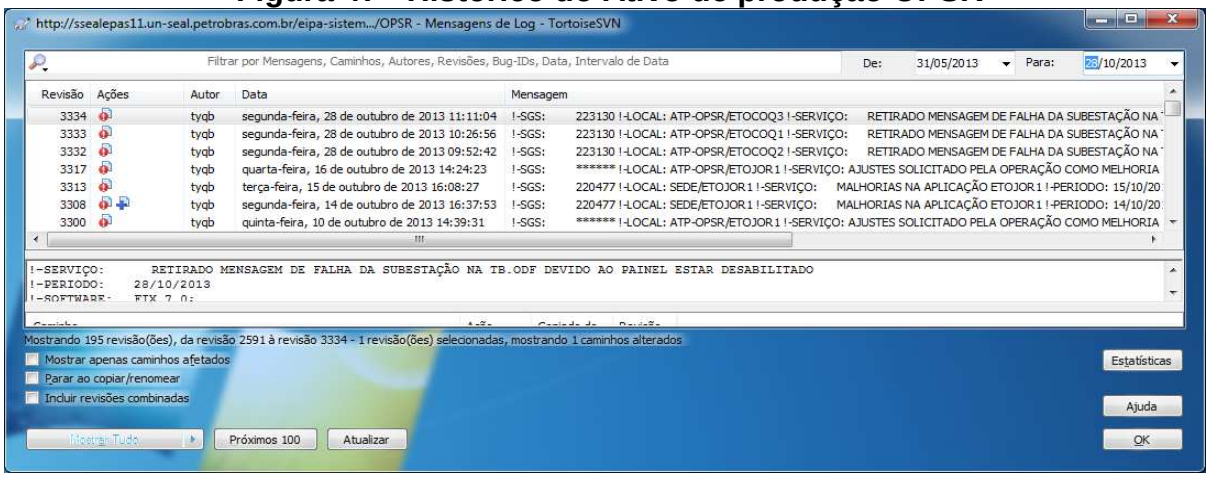
Figura 46 - Estatísticas de Submissão do OPCP



Fonte: Petrobrás.

No ativo de produção OPSR, há no mesmo período, um total de 195 revisões sete delas estão demonstradas na figura 47.

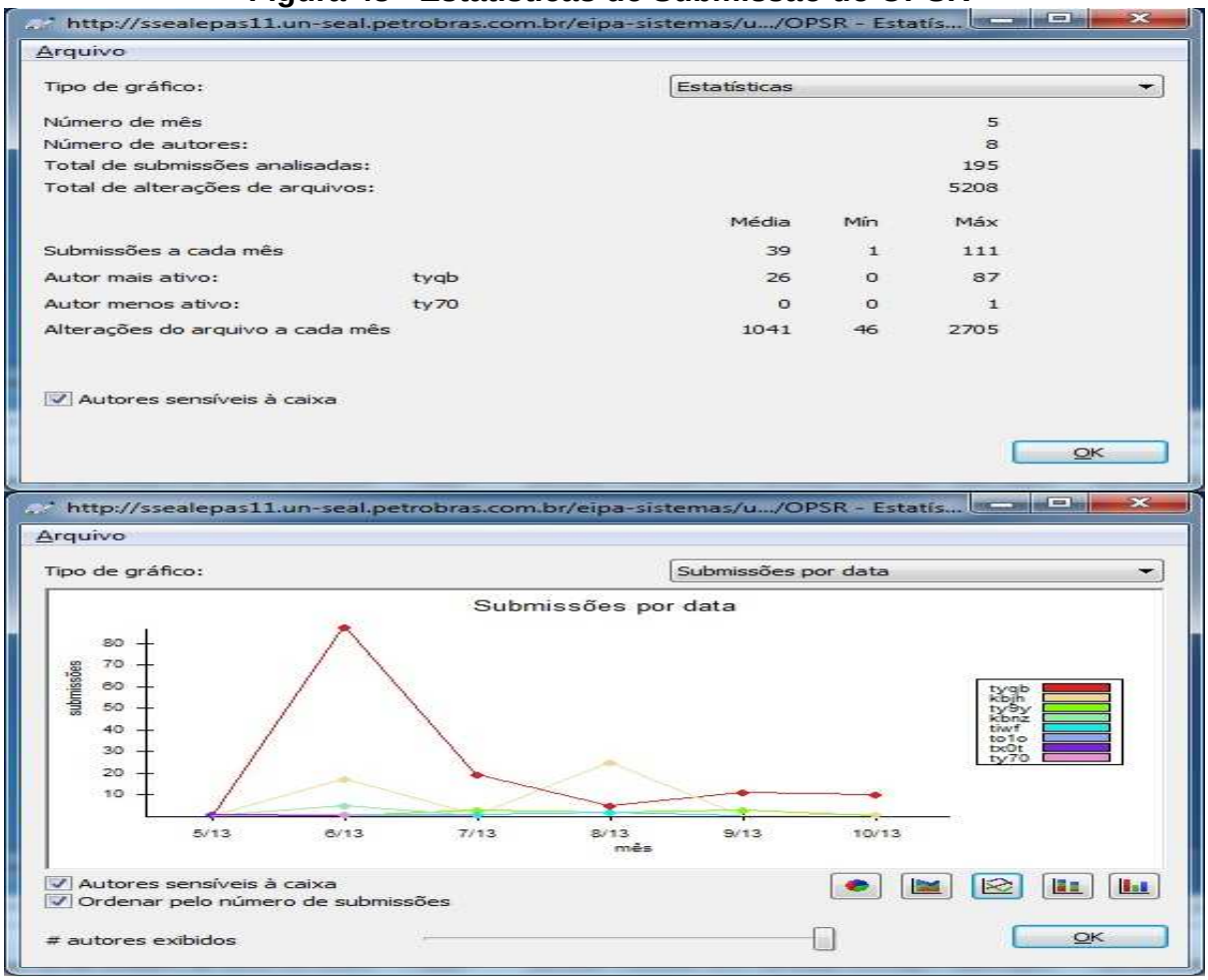
Figura 47 - Histórico do Ativo de produção OPSR



Fonte: Petrobrás.

Oito membros da equipe realizaram estas 195 revisões modificando um total de 5208 arquivos, conforme demonstrados na figura 48.

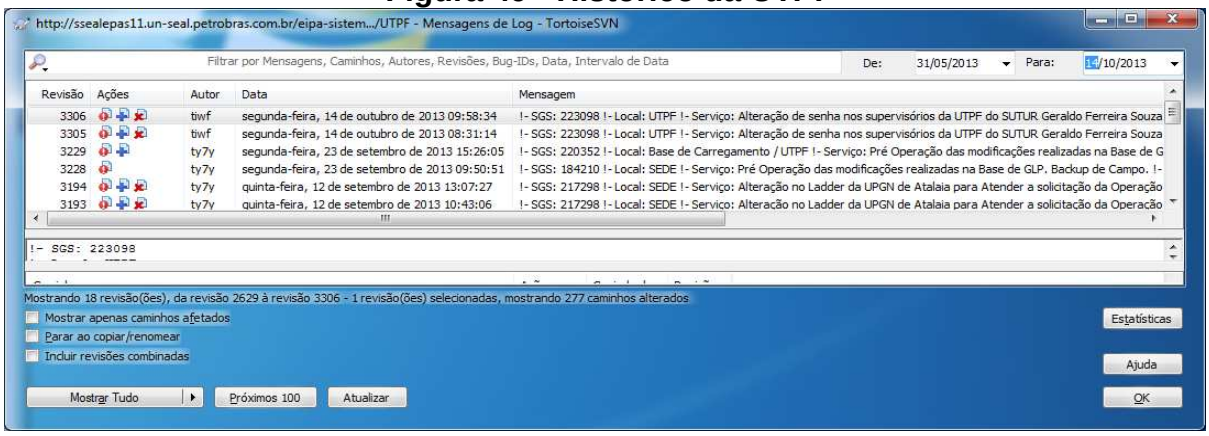
Figura 48 - Estatísticas de Submissão do OPSR



Fonte: Petrobrás.

Neste mesmo período no ativo de produção UTPF, foram realizadas 18 revisões, seis delas estão demonstrados na figura 49.

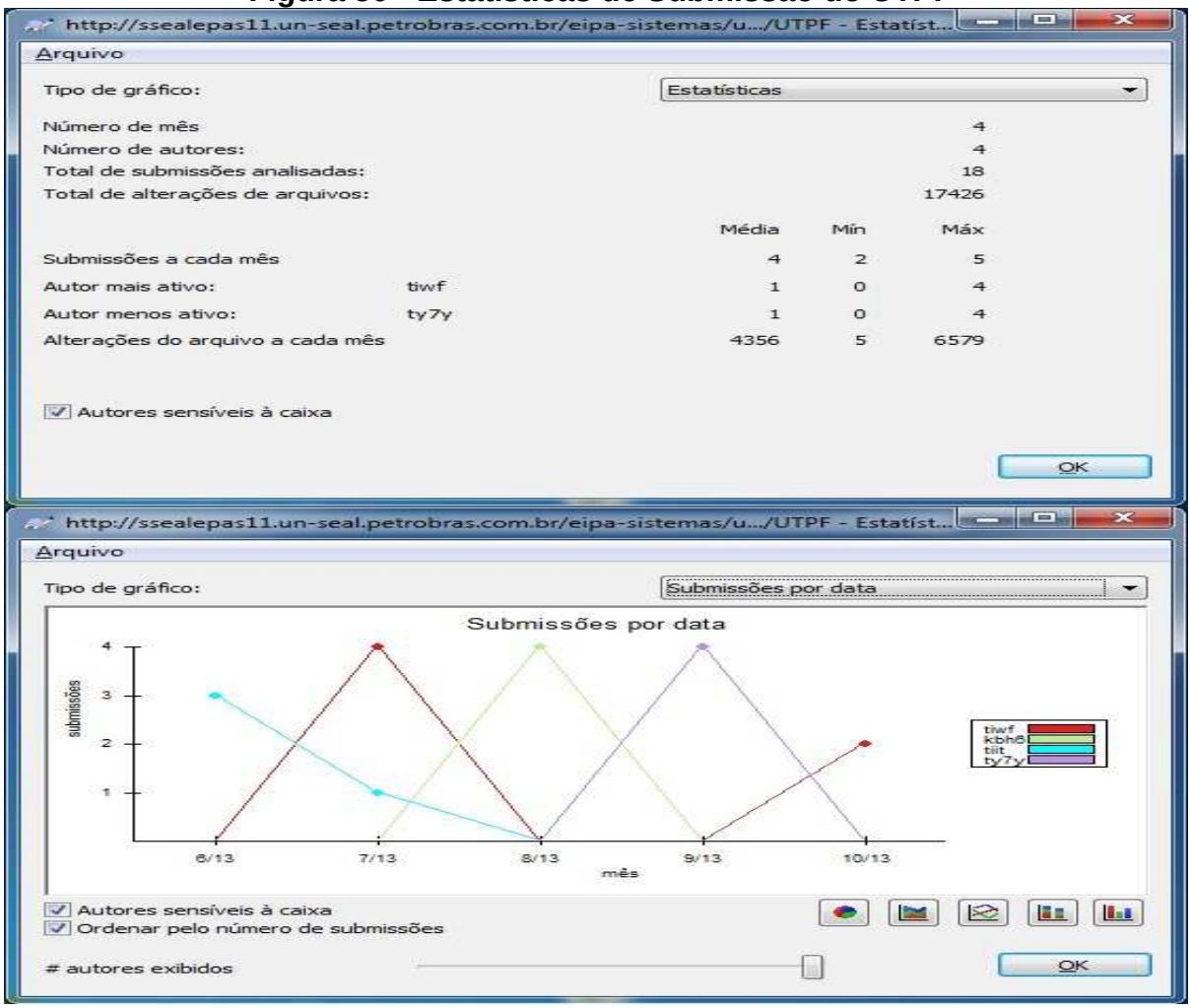
Figura 49 - Histórico da UTPF



Fonte: Petrobrás.

Estas 18 revisões foram realizada por 4 membros da equipe, com um total de 17426 arquivos modificados, conforme demonstrados na figura 50.

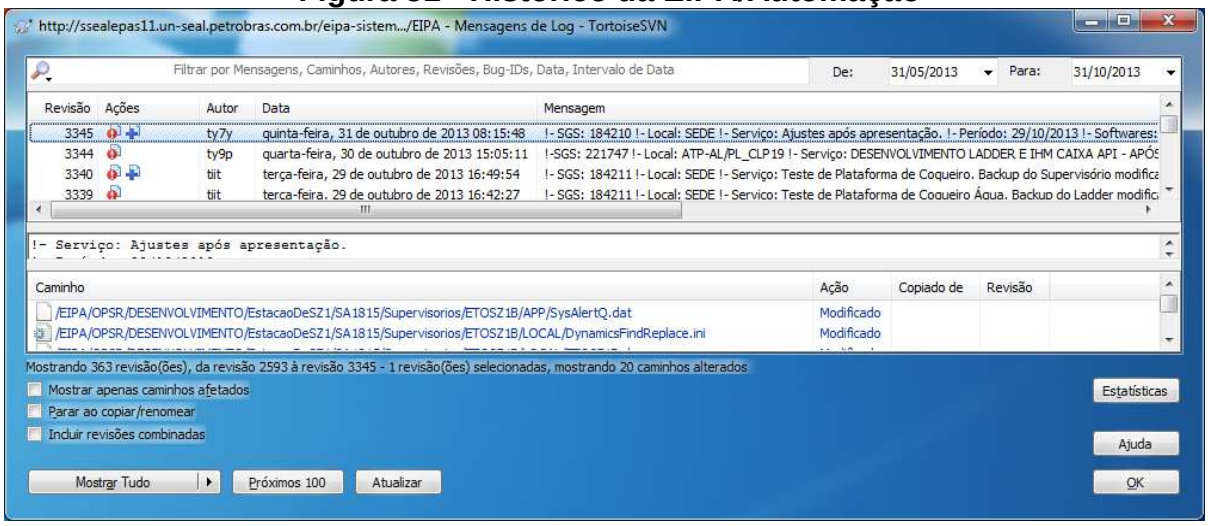
Figura 50 - Estatísticas de Submissão do UTPF



Fonte: Petrobrás.

No ambiente de desenvolvimento da EIPA/Automação, onde ficam localizados os projetos em desenvolvimento, foram realizadas 363 revisões no mesmo período de 31 de maio a 31 de outubro de 2013, quatro delas estão demonstradas na figura 51.

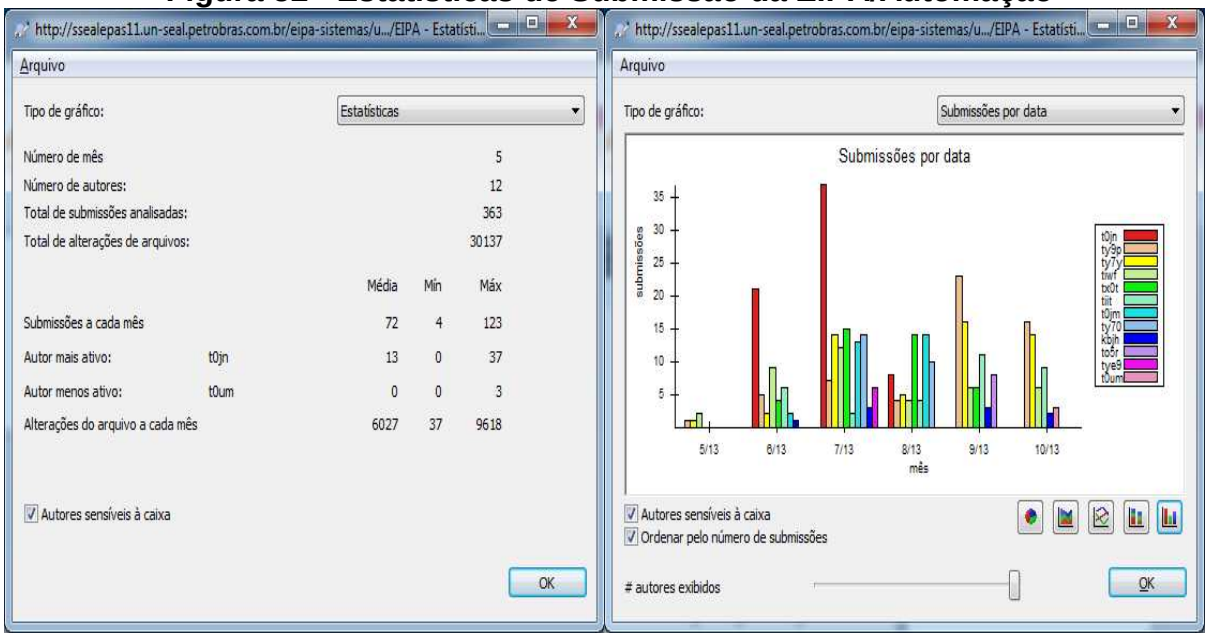
Figura 51 - Histórico da EIPA/Automação



Fonte: Petrobrás.

Doze membros das equipes de desenvolvimento e fiscalização foram responsáveis por estas 363 revisões, que resultaram na modificação de 30137 arquivos, como demonstrado na figura 52.

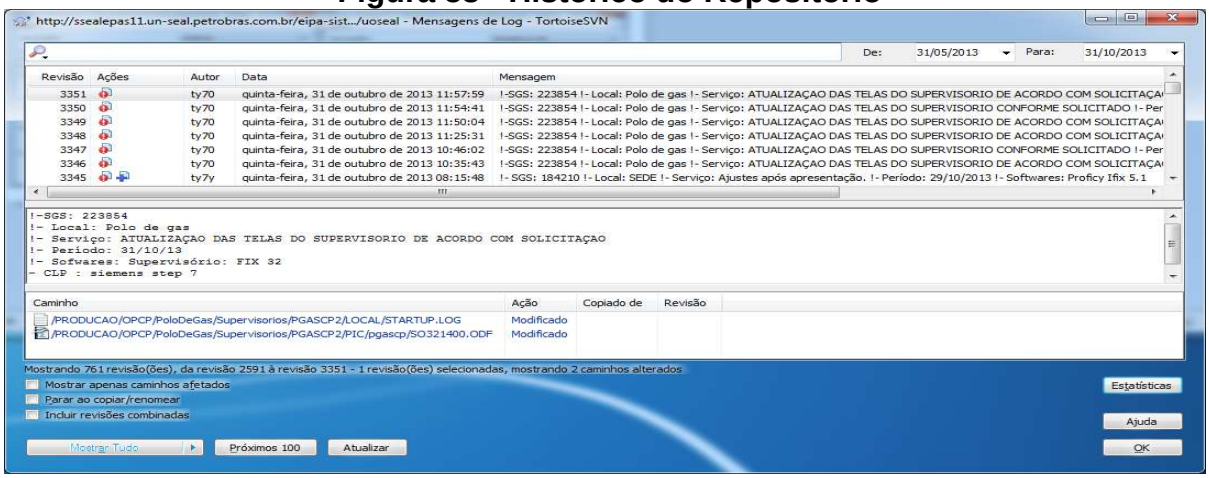
Figura 52 - Estatísticas de Submissão da EIPA/Automação



Fonte: Petrobrás.

Na figura 53, está representado o histórico das ultimas 7 das 761 revisões realizadas em todo o repositório no período de 31 de maio a 31 de outubro de 2013.

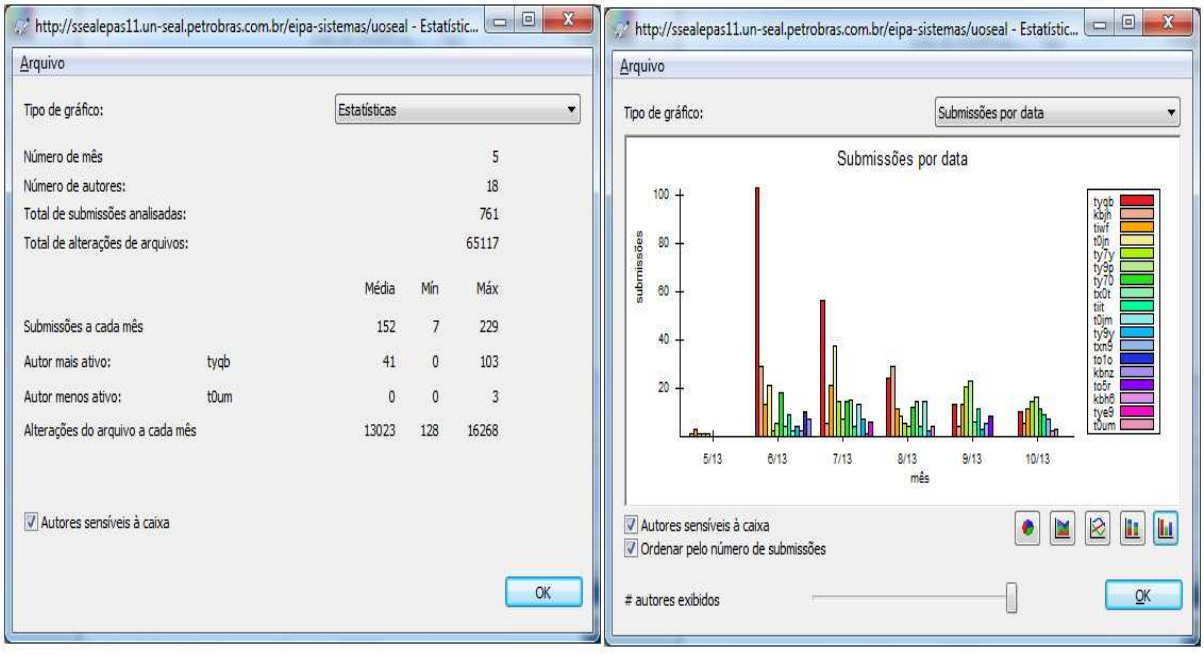
Figura 53 - Histórico do Repositório



Fonte: Petrobrás.

Dezoito membros das equipes com permissão de acesso ao repositório foram responsáveis por 761 revisões no período de 5 meses modificando um total de 65117 arquivos, com uma média mensal de 13023 arquivos modificados, como demonstrado na figura 54.

Figura 54 - Estatísticas de Submissão do repositório



Fonte: Petrobrás

5 CONCLUSÕES

As ferramentas de controle de versão, Subversion e o TortoiseSVN, demonstraram no teste piloto uma boa performance quanto aos tempos de importação, obtenção dos arquivos e um total controle dos itens de configuração submetidos ao repositório, atendendo ao requisito de controle automático de versões.

Devido ao sucesso obtido no piloto de teste, foi recomendado a utilização na rede de automação das ferramentas de controle centralizado Subversion e TortoiseSVN.

Na implementação do Subversion as equipes de desenvolvimento seguiram rigorosamente o plano de ação estabelecido, para garantir o sucesso da implementação.

Para que se tenha um controle das versões eficiente e efitivo dos aplicativos, as equipes envolvidas deverão seguir o plano de gestão.

Com a criação do repositório os aplicativos de automação dos 5 ativos de produção foram colocados sobre o controle do Subversion e todas as versões de qualquer aplicativo se mantiveram integras e confiáveis.

O subversion possibilitou nos ultimo cinco meses o controle de 761 revisões totalizando 65117 arquivos modificados. Demonstrando um ganho significativo em controle para a empresa, visto que o controle via planilha era ineficiente.

REFERÊNCIAS

APACHE SOFTWARE FOUNDATION. **Apache Subversion**: Controle de versão centralizado de classe empresarial para as massas. Disponível em: <http://subversion.apache.org/> . Acessado em: 10 set. 2013.

CHACON, Scott. **Pro GIT.** Disponível em: < http://git-scm.com/book> . Acessado em: 11 set. 2013.

COLLINS-SUSSMAN, Ben; FITZPATRICK, Brian W.; PILATO, C. Michael. **Controle de versão com Subversion:** para subversion 1.4 (compilado da revisão 365). Disponível em: < https://code.google.com/p/svnbook-pt-br/>. Acessado em: 13 set. 2013.

COMUNIDADE BAZAAR. **Bazaar**: O que é Bazaar?. Disponível em: < http://bazaar.canonical.com/en/> . Acessado em: 13 set. 2013.

COMUNIDADE GIT. **GIT**: Disponível em: < http://bazaar.canonical.com/en/>. Acessado em: 13 set. 2013.

COMUNIDADE ECLIPSE. **Resultados de pesquisa da comunidade Eclipse 2013:** Disponível em: <http://eclipse.org/org/press-release/20130612_eclipsesurvey2013.php >. Acessado em: 10 set. 2013.

COMUNIDADE MERCURIAL. **Mercurial**: Gerenciamento de controle de origem: Disponível em: < http://mercurial.selenic.com/about/> . Acessado em: 10 set. 2013.

DIAS, André Felipe. **Subversion.** Disponível em: <http://www.pronus.eng.br/artigos_tutoriais/gerencia_configuracao/subversion.php?pagNum=4> Acessado em: 11 set. 2013.

FALCÃO, Joaquim et al. **Estudo sobre o software livre:** comissionado pelo instituto nacional da tecnologia da informação (iti). Disponível em: <http://www.softwarelivre.gov.br/publicacoes/Estudo_FGV.pdf> . Acessado em: 20 set. 2013.

FERNANDES, Aguinaldo Aragon; ABREU Vladimir Ferra. **Implantando a governança de TI:** da estratégia a gestão de processos e serviços. Rio de Janeiro: Brasport, 2008.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa.** 5. Ed. São Paulo: Atlas, 2010.

HIRAMA, Kechi. **Engenharia de software**: qualidade e produtividade com tecnologia. São Paulo: Elsevier – Campos, 2011.

IT Governance Institute. **Cobit 4.1** Disponível em: < http://www.itgi.org >acessado em: 11 set. 2013.

KÜNG, Stefan ; ONKEN, Lübbe; LARGE, Simon. **TortoiseSVN:** um aplicativo do Subversion para Windows Version 1.8. Disponível em: <http://ufpr.dl.sourceforge.net/project/tortoisesvn/1.8.2/Documentation/TortoiseSVN-1.8.2-pt_BR.pdf> . Acessado em: 20 jul. 2013.

MAGALHÃES, Ivan Luizio; BRITO, Walfrido. **Gerenciamento de serviços de TI na prática:** uma abordagem com base na ITIL. São Paulo: Novatec Editora, 2007
007 livro
MARCONI, Maria de Andrade; LAKATOS, Eva Maria. **Fundamentos de metodologia científica**. 6. ed. 7 reimp. São Paulo: Atlas, 2009.

MICROSOFT. Apresentando o visual SourceSafe. Disponível em: <http://msdn.microsoft.com/pt-br/library/3h0544kx(v=vs.80).aspx> Acesso em: 10 set. 2013

MOLINARI, Leonardo. **Gerência de configuração:** técnicas e práticas no Desenvolvimento do software. Florianópolis: Visual Books, 2007.

MOLINARI, Leonardo. **Teste de software:** produzindo sistemas melhores e mais confiáveis**.** São Paulo: Érica, 2008.

PRESSMAN, Roger S. **Engenharia de software:** uma abordagem profissional. 7. ed**.** Porto Alegre: Artmed, 2011.

RENTES, Miguel. **Subversion:** controlo total sobre software – 2ª parte, Revista Programar, on-line, n.18, fev. 2009. Disponível em: < http://revista-programar.info/?action=editions&type=viewmagazine&n=18>
Acesso em: 15 set. 2013

SANTOS, Antonio Raimundo dos. **Metodologia cientifica**: a construção do conhecimento. 6. ed. Rio de Janeiro: DP&A, 2006.

SIMPLES CONSULTORIA. **Subversion:** lista de comandos. Disponível em: < http://pythonhosted.org/sc.dev.core/vcs/subversion.html>. Acesso em: 15 set. 2013

SOFTEX - Associação para Promoção da Excelência do Software Brasileiro. **MPS-Melhoria de Processo de Software e Serviços:** Guia geral MPS de serviços, 2012. Disponível em:
<http://www.softex.br/wp-content/uploads/2013/07/MPS.BR_Guia_Geral_Servicos_20121.pdf >
Acesso em: 21 out. 2013

SOMMERVILLE, Ian. **Engenharia de software.** 9. ed. São Paulo: Pearson, 2011.

SWEBOK – **Software engineering body of knowledge**. Versão, 2004. IEEE. Disponível em: <http://www.computer.org/portal/web/swebok/html/contentsch7> acesso em: 03 abr. 2013

UBIRAJARA, Eduardo. **Guia de orientação.** Aracaju: FANESE, 2011 (caderno)