UNIFACS UNIVERSIDADE SALVADOR
MESTRADO EM SISTEMAS E COMPUTAÇÃO

RICASSIO CAMILO DE OLIVEIRA SANTOS

DESENHO E MONITORAMENTO DE INSTALAÇÕES DE PRODUÇÃO DE ÓLEO E GÁS ATRAVÉS DA COMPOSIÇÃO DE COMPONENTES DE SOFTWARE

Salvador
2015

RICASSIO CAMILO DE OLIVEIRA SANTOS

DESENHO E MONITORAMENTO DE INSTALAÇÕES DE PRODUÇÃO DE ÓLEO E GÁS ATRAVÉS DA COMPOSIÇÃO DE COMPONENTES DE SOFTWARE

Dissertação apresentada ao Programa de Pós-Graduação em Sistemas e Computação da Universidade Salvador – UNIFACS, Laureate International Universities como requisito parcial para obtenção do título de Mestre.

Orientador: Prof. Dr. Jorge Alberto Prado de Campos.

Salvador
2015

FICHA CATALOGRÁFICA
Elaborada pelo Sistema de Bibliotecas da UNIFACS Universidade Salvador, Laureate International Universities)

Santos, Ricássio Camilo de Oliveira

Desenho e monitoramento de instalações de produção de óleo e gás através da composição de componentes de software./ Ricássio Camilo de Oliveira Santos.- Salvador: UNIFACS, 2015.

82 f. : il.

Dissertação Programa de Pós-Graduação em Sistemas e Computação de UNIFACS Universidade Salvador, Laureate International Universities como requisito parcial à obtenção do título de Mestre.

Orientador: Prof. Dr. Jorge Alberto Prado de Campos.

1. Engenharia de Software – Composição de componentes. 2. Automação. 3. Produção de petróleo e gás. I. Campos, Jorge Alberto Prado de, orient. II. Título.

CDD: 005.1

TERMO DE APROVAÇÃO

RICASSIO CAMILO DE OLIVEIRA SANTOS

DESENHO E MONITORAMENTO DE INSTALAÇÕES DE PRODUÇÃO DE ÓLEO E GÁS ATRAVÉS DA COMPOSIÇÃO DE COMPONENTES DE SOFTWARE

Dissertação aprovada como requisito parcial para obtenção do grau de Mestre em Sistemas e Computação, Universidade Salvador – UNIFACS, Laureate International Universities pela seguinte banca examinadora:

Jorge Alberto Prado de Campos – Orientador _____
Doutor em Spatial Information Science and Engineering, University of Maine at Orono
Universidade Salvador – UNIFACS, Laureate International Universities

André Santanchè _____
Doutor em Ciência da Computação pela Universidade Estadual de Campinas - UNICAMP
Universidade Estadual de Campinas - UNICAMP

Eldman de Oliveira Nunes _____
Doutor em Computação pela Universidade Federal Fluminense - UFF
Universidade Federal Fluminense - UFF

Salvador, 20 de novembro de 2015.

Dedico este trabalho a minha mãe Laurice, a minha esposa Virgínia e a minha filha Clara.

AGRADECIMENTOS

A minha família pelo apoio e compreensão.

A Petrobras, pela oportunidade de realizar esse projeto.

Ao meu orientador Prof. Jorge Campos, pela sua orientação, confiança e paciência para a realização deste trabalho.

Ao Prof. André Santanchè, pelo suporte dado durante o trabalho.

Aos amigos que contribuíram de alguma forma para a finalização deste projeto profissional e pessoal.

RESUMO

As empresas de petróleo e gás estão enfrentando crescentes custos de exploração e produção, fato que somado a atual crise no setor, levam essas empresas a uma necessidade de melhorar continuamente a eficiência de seus processos de produção para se manter competitivas. No que se refere a instalações de produção de petróleo terrestres, uma quantidade significativa de investimento é feito na automação e monitoramento de processos de produção e instalações. Existem poucas empresas especializadas fornecedoras de ferramentas de automação que permitem aos engenheiros e operadores desenvolverem sua própria solução para analisar comportamentos de plantas baseados em dados históricos, monitorar equipamentos e processos em tempo real ou criar uma visão do processo para fins de contabilização da produção. Normalmente, os sistemas que permitem a criação são oriundos da área de automação e tem como características serem sistemas monolíticos, proprietários e de difícil adaptação. Essas características levam empresas maiores, como a Petrobras, a pagar elevados custos de personalização do software proprietário ou desenvolver as suas próprias soluções para lidar com as particularidades de seus processos. Este trabalho apresenta um ambiente de código aberto para modelagem de instalações de produção de petróleo para fins de visualização, monitoramento e contabilização. A solução utiliza Engenharia de Software Baseado em Componentes para superar algumas limitações de soluções de automação proprietárias. Foi desenvolvida uma família de componentes que representam o equipamento, as suas relações, e as regras normalmente encontradas numa instalação de produção de petróleo e gás em terrestre.

Palavras-chave: Automação. PIMS. Componentes de Software. Ambientes de Autoria. Produção de Petróleo e Gás. Processo de Produção de Petróleo.

ABSTRACT

Oil and gas companies are facing increasing exploration and production costs. Therefore, these companies need to continually improve the efficiency of their production processes to stay competitive. Concerning onshore oil production facilities, a significant amount of investment has been made in the automation and monitoring of production processes and facilities. There are few specialized companies that provide authoring systems to allow engineers and operators to develop their own view of the process, to monitor equipment in real-time or to analyze historical data for accounting purpose. Usually, authoring system in automation field are monolithic systems, protected by strict copyright laws, hard to adapt, and difficult to extend. These characteristics lead majors companies, as Petrobras, to pay for the customization of proprietary software or to develop their own solutions to deal with the particularities of their processes. This paper presents an open source authoring environment to model oil production facilities for visualization, monitoring, and accounting purposes. The application use Component-based Software Engineering to overcome some limitations of proprietary automation solutions. We have developed a family of components to represent the equipment, their relationships, and constraints usually found in an onshore oil and gas plant.

Keywords: Automation. PIMS. Software Components. Authoring Enviroment. Oil and Gas Production. Oil and Gas Production Process.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

O processo de contabilização da produção de petróleo depende diretamente da configuração dos elementos envolvidos na operação da produção, desde o poço até o destino final. Assim, quanto maior for o número de variáveis envolvidas, mais complexo se torna o cálculo da quantidade de petróleo produzida por um determinado conjunto de poços. Essa complexidade aumenta principalmente nas áreas com poços onde o petróleo já é extraído a um período grande de tempo, os chamados campos maduros, fato que leva à diminuição da pressão interna do poço e dificulta a saída do fluído (BRAUNS, 2010).

A medição do volume de petróleo e gás produzido é uma atividade de grande interesse nas indústrias produtoras de petróleo e gás. O cálculo deste volume serve como base para o pagamento de impostos e dos royalties de produção. Por esse motivo ele é constantemente auditado pela agência reguladora da exploração de petróleo no Brasil, a ANP, que normatiza os cálculos de volumes produzidos e define os parâmetros que devem ser seguidos pelas empresas produtoras. Por outro lado, as áreas operacionais das empresas produtoras de petróleo necessitam cada vez mais de ferramentas que permitam o controle do processo e o acompanhamento da produção com dados em tempo real. Esse ferramental é fundamental para diminuir as perdas, aperfeiçoar o processo e ter uma visão geral da produção.

Atualmente, existem na Petrobras diversas soluções nas áreas de TI e de automação para o monitoramento, o acompanhamento da produção e para o cálculo de volume produzido. Desta forma, o processo de contabilização da produção está distribuído entre diversas aplicações produzidas em áreas diferentes da empresa. A maior parte das aplicações de acompanhamento de processo é desenvolvida pela área de engenharia e operação. As atividades de monitoramento da produção, por exemplo, são baseadas ferramentas proprietárias, monolíticas e com pouca ou nenhuma flexibilidade para serem acopladas com as demais ferramentas disponíveis na empresa. Já as aplicações que tratam do cálculo da produção são desenvolvidas, notadamente, pela área de TI. Esta realidade gera um grande número de softwares replicados e desconectados entre si, dificultando a visualização e o entendimento do processo como um todo e a integração das aplicações existentes.

Uma possível solução para essa fragmentação de soluções tratando de problemas correlatos ou complementares seria a criação de uma infraestrutura de

software que permitisse o monitoramento da produção e a documentação do processo para fins de contabilização. Essa infraestrutura deve possuir um ambiente autoral onde usuários sem experiência em desenvolvimento de software possam mapear seus processos com componentes que representassem elementos do processo de produção; oferecer uma visão gráfica do processo e do fluxo da matéria prima; utilizar informações obtidas tantos dos sensores quantos dos sistemas operacionais utilizados nos campos de produção de forma transparente para os desenvolvedores; e permitir que técnicos da área operacional, de engenharia e de TI possam trabalhar de forma colaborativa e complementar.

Este trabalho propõe a criação de um ambiente de autoria que permita a criação de plantas para monitoramento em tempo real ou desenho de fluxos de produção para fins de controle e documentação. A aplicação proposta é baseada em componentes de software, aberta para desenvolvimento colaborativo e flexível para integração com qualquer sistema ou fontes de dados de interesse.

1.1 JUSTIFICATIVA

Este trabalho foi construído com base na realidade operacional da Petrobras, empresa patrocinadora deste trabalho. O autor atua no desenvolvimento e manutenção de aplicações voltadas para área de automação na Petrobras com foco no processo de produção de petróleo e gás em instalações terrestres.

A Petrobras possui um sistema de confecção de plantas do processo que é utilizado rotineiramente pelos engenheiros e operadores, o PI ProcessBook desenvolvido pela OSIsoft (OSI SOFTWARE INC, 2009). Esse sistema faz parte de um pacote de ferramentas oferecido pela Empresa desenvolvedora da solução com sua licença de utilização atrelada à aquisição de outras soluções na área de automação. A integração desse sistema com os demais sistemas corporativos da Petrobras é feita de maneira rudimentar e ainda muito complexa para os usuários sem experiência em desenvolvimento de software.

Frequentemente os usuários do ambiente de autoria da OSIsoft se deparam com barreiras para modelar o processo de produção de uma determinada instalação física ou lógica. Para o desenho de uma Estação de Produção com a obtenção de informações em tempo real, por exemplo, todos os locais de medição e tratamento de petróleo são mapeados para construção do fluxo, porém nem todos os pontos de

medição existentes na estação estão automatizados. Em determinados locais, especialmente em instalações de poços maduros, a extração dos dados é feita manualmente e alimentada diretamente pelo operador no sistema que contabiliza a produção de petróleo. Para que o PI ProcessBook acesse um dado que não esteja automatizado, mas que esteja disponível em bancos de dados relacionais, é necessário um esforço de programação avançada para configurar essa interface. Além de conhecimentos de programação também é necessário entender do modelo de dados para a construção de consultas SQL que retornem a informação desejada da base de dados operacional.

1.2 OBJETIVOS

O objetivo deste trabalho é o desenvolvimento de um ambiente que permita a criação de plantas industriais para monitoramento da produção de petróleo e gás através da composição de componentes que possibilitem, de maneira transparente, o desenho do processo de negócio da planta mapeada.

Os modelos composição de componentes se referem a plantas industriais em funcionamento e irão permitir o acompanhamento e controle dos diversos equipamentos instalados nas unidades de produção da Petrobras. Para o desenvolvimento do ambiente de composição para o monitoramento de plantas industriais de produção e tratamento de petróleo será utilizada uma abordagem baseada em componentes de software. Esta abordagem irá permitir o desenvolvimento de uma aplicação escalável e facilmente adaptável às necessidades de uma empresa do porte da Petrobras, que possui uma grande diversidade no seu parque industrial. Desta forma, este trabalho possui os seguintes objetivos específicos:

a) Identificação de um modelo de componente de software que possibilite a representação dos equipamentos;

b) Definição de uma taxonomia de equipamentos de acordo com as características do equipamento e a etapa do processo onde ele é necessário;

c) Definição de uma família de componentes de software que permita a representação de equipamentos industriais. Os referidos componentes deverão incorporar a representação gráfica dos equipamentos de medição e todo código

responsável pela conexão com o sistema de automação (sensores) e com dados históricos armazenados em banco de dados.

d) Definição de uma família de componentes que modele o fluxo do processo da instalação através da agregação dos componentes de equipamentos mencionados anteriormente. Os componentes de processo permitirão a definição das regras para composição dos componentes de equipamentos, explicitando as ligações possíveis entre equipamentos, a ordem dos equipamentos no fluxo do processo industrial e a representação das associações definidas no contexto do processo.

1.3 METODOLOGIA

Inicialmente foi realizada uma pesquisa na Petrobras, unidade da Bahia, sobre as ferramentas disponíveis para criação de plantas industriais e documentação do processo de produção. As aplicações encontradas que permitem o monitoramento de plantas são todas da área de automação. A documentação do processo que permite a rastreabilidade da contabilização da produção é atualmente realizada utilizando planilhas eletrônicas.

Em seguida foi realizada uma pesquisa sobre a ferramenta de automação que é utilizada para criação de sistemas que representam plantas de processo para fins de monitoramento. O fluxo dos dados foi mapeado desde o nível mais baixo, no equipamento instalado no campo, até a disponibilização para o usuário que constrói o monitoramento.

Para permitir uma representação ampla e extensível dos equipamentos e das plantas de processo foi identificada a necessidade de utilizar o conceito de componentes de software. Dessa forma foi realizada uma revisão bibliográfica sobre componentes de software e modelos de componentes para identificar um modelo de componente que permitisse liberdade de implementação e fosse adequado as necessidades especificas desse trabalho na área de automação.

Definido o modelo de componentes foi realizada uma pesquisa dos trabalhos realizados utilizando o modelo de componente adotado. Identificamos um ambiente de autoria na área de laboratório e um *template* para modelar composição de componentes também na área de laboratório.

A partir do modelo de componentes adotado e da implementação deste modelo no ambiente de autoria para criar laboratórios na Web, foi criado um conjunto de componentes para representar os equipamentos utilizados no processo de produção de petróleo e gás e seus relacionamentos com os diversos repositórios disponíveis na Petrobras. Um conjunto de componentes para modelar o processo de produção foi proposto como uma evolução ao *template* para representar as composições de componentes. O conjunto de componentes de processo foi utilizado para representar um tipo de instalação de produção de petróleo e gás.

Para efeito de validação e prova de conceito foi desenvolvido um conjunto de componentes de equipamentos e de componentes de fluxo para efeito de criação de uma planta de monitoramento de uma instalação típica da Petrobras.

1.4 CONTRIBUIÇÕES

As contribuições desse trabalho dizem respeito a componentes de software e o desenvolvimento de sistemas na área de petróleo e gás. Nesse contexto a principal contribuição deste trabalho está na evolução do conceito de Metalaboratórios para representação de processos industriais, na criação de uma família de componentes para representar equipamentos utilizados na produção de petróleo e gás e de componentes para representação do processo.

Para a Petrobras a principal contribuição desse trabalho é a disponibilização de uma alternativa às das ferramentas proprietárias atualmente existentes para a criação de monitores de produção. Além da criação dos monitores de produção, os componentes propostos permitem a documentação do processo para fins de contabilização da produção ou auditoria do processo pela agência reguladora.

1.5 AUDIÊNCIA

Embora este trabalho tenha como foco principal a indústria petroquímica, ele tem como audiência um público bastante diversificado, pois reúne vários temas e contribuições na área de monitoramento e controle de processo, componentes de software e ambientes de autoria.

Desta forma acreditamos que o público deste trabalho englobe estudantes e profissionais da área de automação e computação, engenheiros de produção,

operadores de produção, analistas de sistemas, analistas e técnicos da agencia reguladora de petróleo ou qualquer pessoa interessada em questões referentes a construção de ferramentas para monitorar um processo de produção.

1.6 APRESENTAÇÃO

O restante dessa dissertação está organizada da seguinte forma: No capítulo dois é feita uma contextualização da infraestrutura de automação existente nas indústrias abordando o fluxo de dados entre as camadas definidas para a área de sistemas de automação. Neste capitulo é discutido, também, os trabalhos que tratam de ambientes de autoria para criação e execução de componentes de software na Web.

O capítulo três apresenta os componentes de software desenvolvidos para representar os instrumentos utilizados no processo de produção de petróleo e gás. Para possibilitar essa representação é proposta uma classificação dos instrumentos utilizados no processo de produção de óleo e gás de acordo com a etapa de sua utilização.

O capítulo quatro apresenta componentes de software desenvolvidos para representar os diversos processos de produção de petróleo e gás através da composição de componentes. O capítulo cinco apresenta o ambiente de composição de componentes WebPlantas, criado como prova de conceito dos componentes propostos.

O capítulo seis apresenta as conclusões e indica as possíveis extensões para trabalhos futuros.

2 AUTOMAÇÃO E COMPONENTES

A automação representa o principal meio de acompanhamento do processo de produção de petróleo e gás. Na Petrobras, todas as aplicações que têm como objetivo efetuar o cálculo da produção de petróleo ou monitorar e controlar a produção possuem algum vínculo com as soluções de automação. Quanto mais estreito for o vínculo da aplicação com o sistema de automação, mais confiáveis serão as informações geradas, pois menor será a interferência humana.

Neste capítulo é feita uma contextualização da infraestrutura de automação existente na indústria de Petróleo, utilizando como referência a Petrobras. O foco são os equipamentos utilizados no processo de produção de petróleo, descrevendo o fluxo dos dados desde os sensores até os sistemas historiadores de dados. As representações dos equipamentos utilizados no processo de produção de petróleo são discutidas neste que capítulo, também a proposta deste trabalho de representar os equipamentos como componentes de software. Neste capítulo é discutido, também, o modelo de componente utilizado como referência para os componentes propostos, assim como os ambientes de autoria baseados em componentes que servem de base para a proposta desse trabalho.

2.1 AUTOMAÇÃO

A tecnologia de automação é uma ferramenta poderosa para a obtenção da garantia da qualidade de processos, bem como visa uma otimização da rotina operacional que pode ser traduzida em aumentos de produtividade e redução de custos operacionais (TSUTIYA, 2004; VIANNA, 2008). O termo automação descreve um conceito amplo, envolvendo um conjunto de técnicas de controle utilizadas para criação de um sistema ativo, capaz de fornecer a melhor resposta em função das informações que recebe do processo em que está atuando (SENAI-PE, 2008). A automação pode ser definida também como um sistema, que apoiado por computador ou equipamento programável, remova o trabalhador de tarefas repetitivas e que vise a soluções rápidas e econômicas para atingir os objetivos das indústrias (BOARETTO, 2008).

A forma mais utilizada na Petrobras para visualização e acompanhamento da produção de petróleo é através de ferramentas oriundas de soluções especificas de automação. Estas ferramentas evoluíram para sistemas grandes e complexos, que apesar de permitir uma visão gerencial a partir de dados obtidos direto dos equipamentos utilizados no campo, falham em termos de integração e flexibilidade, pois são construídas com base em sistemas de automação proprietários e inflexíveis.

As ferramentas de visualização e acompanhamento surgiram a partir da necessidade de utilização dos dados do processo nos níveis mais altos das empresas como subsidio para a tomada de decisão. Elas são construídas no topo das soluções de automação para controle do processo e são classificados como Sistemas de Gerenciamento de Informações da Planta ou PIMS (do inglês, Plant Information Management System). Na Figura 1 temos uma adaptação da pirâmide de automação representando os diversos níveis de gerência de informação relacionados aos processos de automação na Petrobras.

Figura 1 – Níveis de gerência de informação de processos de automação na Petrobras

Fonte: Autoria própria do autor desta dissertação (2015).

O nível de Supervisão e Controle (Figura 1) compreende o núcleo do sistema de automação. Neste nível são coletados os dados e definidos as ações de controle.

Com a evolução dos equipamentos utilizados no campo, uma grande variedade de dados proveniente destes pôde ser disponibilizada para outras aplicações e, atualmente, o grande desafio da automação industrial é transformar esse grande volume de dados em informação útil. Este cenário favoreceu o

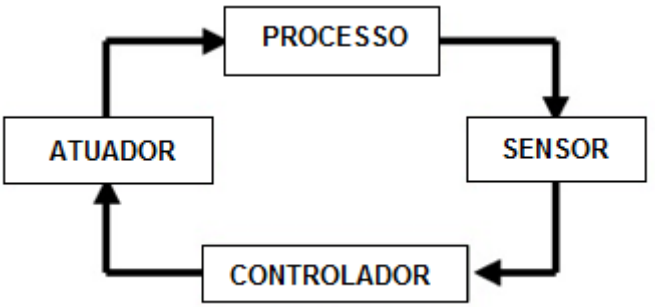
aparecimento de aplicações PIMS, pois elas permitem que os usuários utilizem uma grande variedade de dados para as mais diversas finalidades.

Um dos objetivos deste trabalho é disponibilizar para os usuários outros meios de utilização e obtenção dos dados oriundos das soluções de automação, inclusive daquelas já armazenadas no PIMS, como forma de produção de informação utilizando ambientes colaborativos, flexíveis e abertos. Dessa forma, é imprescindível o entendimento do fluxo de dados nas soluções de automação, desde os sensores até o armazenamento nas bases de dados temporais dos PIMS.

Como forma de possibilitar o controle e acompanhamento do processo de produção, por meio de processos automáticos, é necessário que existam dispositivos que coletem dados e os disponibilizem para consumo de outros dispositivos ou aplicações. Os dispositivos, que são os instrumentos de medição e controle de processos, são estudados e aplicados pela área de conhecimento da engenharia denominada instrumentação (GOMES, 2009). Esta área de conhecimento é responsável pela medição das variáveis do processo, o que, além de outras responsabilidades, diz respeito a converter grandezas físicas como nível, pressão, vazão e quantidade de água e sedimentos no petróleo, em sinais elétricos padronizados, ou seja, corrente, tensão ou resistência ôhmica (TSUTIYA, 2004).

Um importante elemento da malha de controle é o sensor (Figura 2). O sensor é o responsável pela interface do sistema de controle com o processo real (PAIOLA; TEIXEIRA, 2011; SILVEIRA; SANTOS, 1998). Os atuadores são dispositivos que interferem no processo e determinam uma ação física, por meio de uma requisição do sistema controlador, denominada ação de controle. São exemplos de atuadores: válvulas e cilindros pneumáticos, válvulas proporcionais, motores, aquecedores, entre outros (SILVEIRA; SANTOS, 1998). Enquanto os sensores captam informações sobre o processo, os atuadores interferem neste mesmo processo, gerando assim, o controle (Figura 2) (TSUTIYA, 2004; ZAMPRONHA, 2006).

Figura 2 – Diagrama simplificado do funcionamento de um sistema de controle



Fonte: Autoria própria do autor desta dissertação (2015).

As aplicações que integram os elementos do sistema de automação com o propósito de monitorar e operar parte ou todo processo são denominadas sistemas SCADA - Sistemas de Supervisão, Controle e Aquisição de Dados (do inglês, *Supervisory Control and Data Acquisition*) (FONSECA, 2007). Esses sistemas coletam dados do processo através de dispositivos remotos, principalmente os Controladores Lógico Programáveis (CLP), formatam estes dados e os apresenta ao operador em uma multiplicidade de formas. O objetivo principal dos sistemas SCADA é propiciar uma interface de alto nível do operador com o processo, com informações de todos os eventos de importância da planta. (GOMES, 2009). Apesar de propiciar uma visão completa e de alto nível da planta, os sistemas SCADA permitem a visualização de parte do processo e com histórico limitado. Para visualização mais abrangente de todo o processo e utilização de dados históricos são utilizadas as aplicações PIMS discutidas na próxima seção.

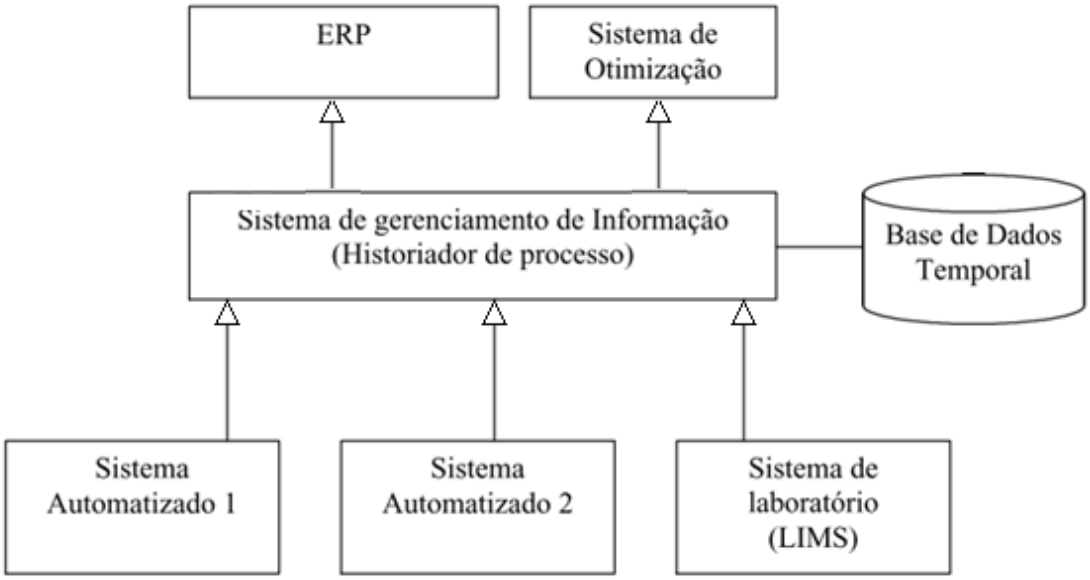
2.2 PIMS E PROCESSBOOK

Os sistemas PIMS estão no centro da pirâmide da automação (Figura 1), estabelecendo a ponte entre os sistemas de gestão e as soluções de automação. Os sistemas PIMS definem as aplicações que adquirem dados de processo de diversas fontes, os armazenam em uma base de dados temporal e os disponibilizam através de diversas formas de representação (SEIXAS FILHO, 2011). O PIMS nasceu na indústria petroquímica para resolver problemas da fragmentação de dados e

proporcionar uma visão unificada do processo. Esta ferramenta é fundamental para o operador e o engenheiro de processos, pois a partir de uma estação de trabalho pode-se visualizar tanto os dados em tempo real como históricos dos equipamentos da planta. Através destes sistemas é possível montar tabelas, gráficos de tendência e sinópticos e eliminar as ilhas de informação, concentrando em uma única base de dados a informação sobre todos os aspectos de uma planta (CARVALHO; FONSECA; FILHO, 2005).

A capacidade de gerar outros dados através de cálculos e de armazená-los por longos períodos de tempo constitui um grande ganho para o analista de processos que passa a gerar relatórios sem ter que se preocupar se o dado se origina em um CLP ou sistema SCADA. Os historiadores de processo encapsulam os dados e torna transparente para o usuário a origem dos dados, abstraindo toda a complexidade dos sistemas de automação que estão nos níveis mais baixos de automação. A Figura 3 apresenta uma visão simplificada do fluxo dos dados que alimentam o banco de dados temporal de um PIMS e os principais consumidores de seus dados. As caixas localizadas abaixo do historiador de processo representam as fontes que alimentam a base de dados temporal, que são os sistemas automatizados e os sistemas de laboratório. Já as caixas que estão desenhadas acima do historiador de processos representam os principais sistemas consumidores dos dados armazenados, que são os sistemas de otimização e os sistemas integrados de gestão - ERP (do inglês, *Enterprise Resource Planning*).

Figura 3 – Fluxo de informação nos sistemas historiadores

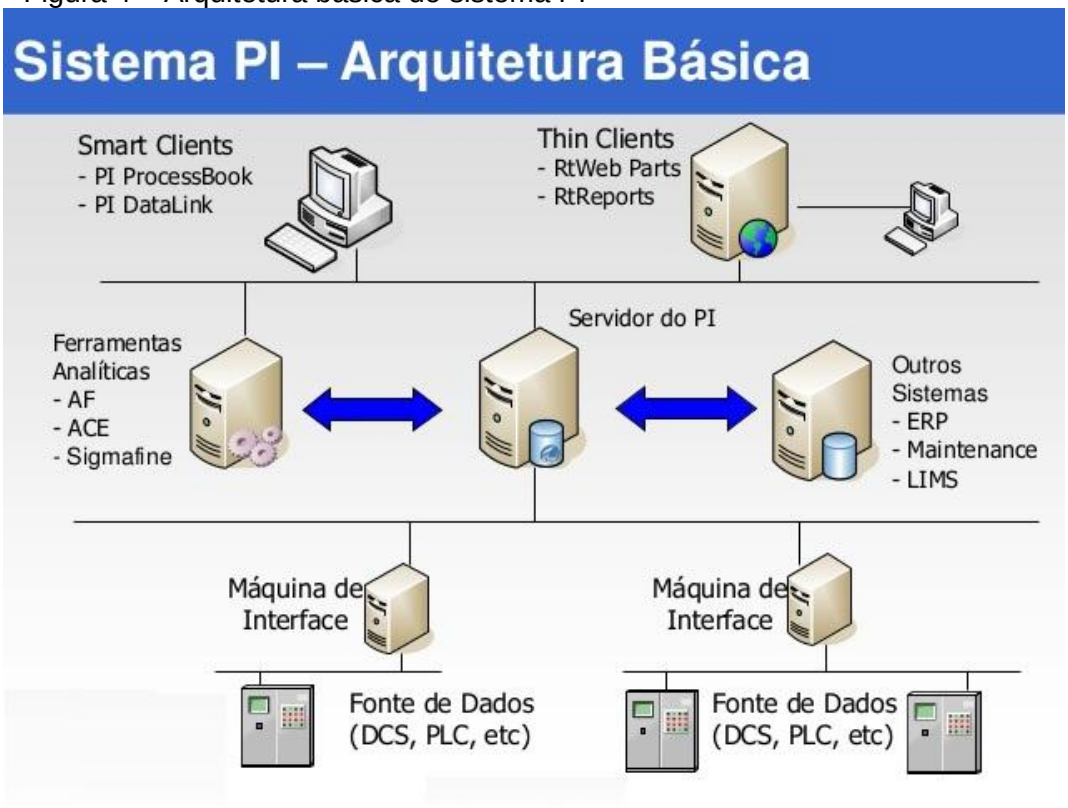


Fonte: Seixas Filho (2011).

Uma das principais empresas fabricantes de PIMS é a *OSIsoft* com seu produto o PI (*Plant Information*) (CARVALHO; FONSECA; FILHO, 2005). A Petrobras, assim como outras empresas do setor, adota o PI como principal solução PIMS para acompanhar seus processos. O sistema PI corresponde a um conjunto de módulos de software servidor/cliente para monitoramento e análise de plantas de processo (Figura 4). O servidor do PI, também chamado de PI *Universal Data Server* (PI UDS), é o núcleo deste sistema, atuando como servidor de dados baseado em Microsoft Windows (SCHEUER, 2004). Na arquitetura básica do sistema PI as máquinas de interface são responsáveis por coletar os dados das diversas fontes e transmiti-los para o servidor PI, que disponibiliza as informações coletadas através de diversas aplicações, a exemplo do *ProcessBook*.

De forma a permitir integração com outros sistemas, o sistema PI disponibiliza uma biblioteca de funções que permitem ler e escrever valores no servidor PI, além de alterar ou obter configurações de pontos (SCHEUER, 2004). Essa biblioteca denominada PI-SDK permite que sistemas desenvolvidos para a plataforma Windows possam ter acesso a dados historiados na base de dados do sistema PI. A biblioteca PI-SDK é disponibilizada através da aquisição de licenças de utilização junto a empresa fabricante.

Figura 4 – Arquitetura básica do sistema PI



Fonte: OsiSoft OSI SOFTWARE INC (2009).

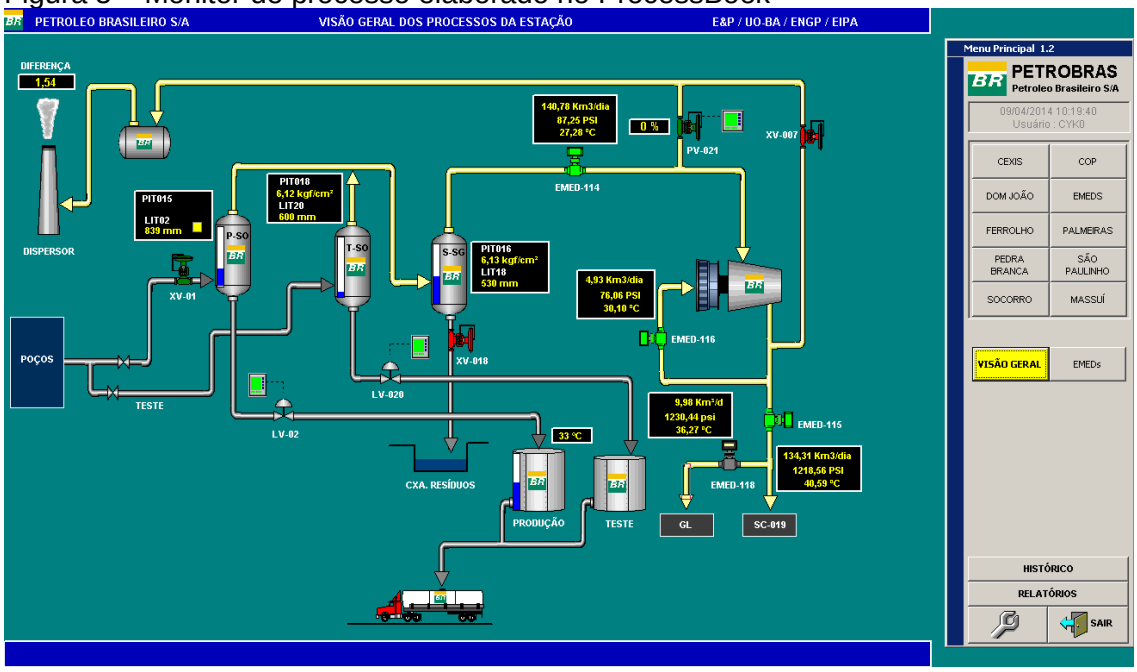
Dentre a suíte de aplicações fornecidas pela OSI Software e de especial interesse nessa dissertação encontra-se o *ProcessBook*. O *ProcessBook* é a principal ferramenta utilizada na Petrobras para confecção de plantas para acompanhamento do processo. Esse software, que possui elementos de um ambiente de autoria que permite uma relativa independência dos engenheiros e operadores para criação de telas customizadas para o acompanhamento de suas atividades no controle do processo.

O *ProcessBook* disponibiliza um banco de elementos que representam equipamentos do processo de produção de óleo e gás e elementos gráficos para representar o deslocamento do fluido entre eles. Os elementos podem ser associados a identificadores do sistema de automação que mapeiam variáveis do processo no sistema PI (OSI SOFTWARE INC, 2009). Os recursos de autoria do *ProcessBook* permitem que usuários montem o esquema da planta que desejam monitorar de forma simples e intuitiva.

A Figura 5 ilustra uma planta modelada no *ProcessBook* mapeando um processo de produção da Petrobras, onde os equipamentos exibidos estão na

biblioteca da aplicação e os quadros negros ao lado de cada equipamento são valores obtidos em tempo real do sistema de automação. Os principais equipamentos mostrados na Figura 5 são tanques de óleo e gás e separadores de fluidos, que são utilizados para separar óleo e gás (separadores bifásicos) ou óleo, gás e água (separadores trifásicos). Neste trabalho, a representação gráfica dos equipamentos de uma planta ou parte dela que apresente dados obtidos do sistema de automação é denominada de Monitor de Processo. A criação de um ambiente de autoria baseado em componentes de software para criação de Monitores de Processo é o principal objetivo desta pesquisa.

Figura 5 – Monitor de processo elaborado no ProcessBook



Fonte: Livro de processo da Petrobras.

Um dos principais problemas encontrados por usuários do *ProcessBook* é incluir no Monitor de Processo os equipamentos que não estão ligados à rede de automação. Diversas instalações da Petrobras, especialmente nas áreas com poços maduros, possuem grande parte de seus equipamentos não automatizados. Isso ocorre em virtude das dificuldades de implementação de automação nessas áreas ou por decisão gerencial frente aos custos inerentes a implementação e manutenção de áreas automatizadas.

Nos casos, onde os equipamentos não estão ligados à rede de automação, os dados de produção são coletados manualmente pelos operadores e inseridos em

sistemas operacionais, que armazenam os dados em bancos de dados relacionais. A aplicação *ProcessBook* permite que dados provenientes de bancos relacionais de outras aplicações sejam mapeados e inseridos nas plantas desenhadas no sistema. Esta ação, entretanto, demanda conhecimentos avançados dos usuários em linguagens de programação e principalmente no modelo de dados de onde a informação será obtida. Este tipo de conhecimento é essencial para a definição de consultas SQL que irão produzir a informação necessária para construção do monitor de processo da planta.

Outra característica do *ProcessBook*, que é comum nas aplicações similares disponíveis no mercado, é a dificuldade de integrar essa ferramenta com os demais sistemas existentes na empresa. Esses softwares possuem código fechado e as áreas de TI das empresas clientes não possuem acesso para realizar adaptações na aplicação. Com isso é necessário utilizar as diversas soluções disponibilizadas pelas empresas fabricantes para efetuar a integração do software com os demais sistemas da empresa. Essa característica tornam as empresas clientes dependentes das empresas fabricantes.

Uma característica positiva do ProcessBook, especialmente do ponto de vista dos operadores e engenheiros, é o fato desse software possuir elementos de um software de autoria. Nesse ambiente operador/engenheiro atua como autor, construindo ou editando os monitores de processo que serão disponibilizados no repositório do sistema PI. Este repositório é denominado de Livro de Processo.

Um ambiente de autoria para criação de monitores de processo baseado na composição de componentes de software é uma alternativa para integração dos diversos sistemas existentes na Petrobras. Esta integração deverá permitir a criação de plantas industriais para monitoramento da produção utilizando qualquer fonte de dados existente na empresa e que possam ser executadas independentemente de plataformas tecnológicas proprietárias. A solução proposta também deve permitir a reutilização das soluções desenvolvidas pelos engenheiros e operadores como base para o desenvolvimento de soluções mais complexas sem que essas necessitem ser alteradas para isso. Uma das formas de atingir esse objetivo é a representação dos elementos que compõem uma planta como componentes de software. Os componentes de software permitem o encapsulamento de qualquer tipo de conteúdo, além de permitir a criação de aplicações através da composição dos componentes desenvolvidos previamente. Na próxima seção serão discutidas

algumas iniciativas na área de componentes de software e de sistemas de autoria baseados em componentes de software.

2.3 COMPONENTES DE SOFTWARE

Os Componentes de Software são pacotes independentes de software que provêm funcionalidades através de interfaces bem definidas (HOPKINS, 2000). Em D´Souza e Wills (1998) os componentes de software são definidos como um pacote coerente de artefatos de software que podem ser desenvolvidos de forma independente, entregues como uma unidade e que podem ser compostos, inalterados, com outros componentes para construir algo maior. Similarmente, Szyperki (1998) define componente de software como uma unidade de composição com interfaces contratualmente especificadas e com dependências apenas contextuais, podendo ser construída de forma independente e estando sujeita a composição por terceiros.

Em Councill (2007) o termo componente de software é definido através da composição de três elementos: 1) o componente de software propriamente dito; 2) o modelo de componente e 3) a infraestrutura de componente de software. O componente de software é um elemento de software coerente com o modelo de componente, podendo ser desenvolvido de forma independente e composto com outros componentes sem necessidade de modificações, mas em conformidade com o padrão de composição. O modelo de componente define as formas específicas de interações e os padrões de composição de componentes. A implementação do modelo do componente se refere ao conjunto de elementos de software executáveis necessários para apoiar a execução de componentes que estejam em conformidade com o modelo. Finalmente, a infraestrutura de componente de software representam o conjunto de componentes de software que interagem para garantir que subsistemas construídos utilizando estes componentes e suas interfaces irão satisfazer claramente as especificações de performance. Estas três definições mostram a importante relação entre a infraestrutura de componente de software, os componentes de software e o modelo de componente.

Nas definições apresentadas a principal característica dos componentes de software fica evidenciada, que é a utilização destes para compor estruturas maiores. Um ramo da Engenharia de Software, a chamada Engenharia de Software

Baseada em Componentes (ESBC) pretende construir sistemas complexos a partir de componentes menores, unidos por um software que cria uma única forma e comportamento do sistema básico (COX; BAOMING SONG, 2001). Idealmente, um novo sistema pode ser construído utilizando, em sua maioria, partes predefinidas, com apenas um pequeno número de novos componentes necessários (CRNKOVIC; STAFFORD; SZYPERSKI, 1968).

Segundo Hopkins (2000) existe alguns cuidados que devem ser observados com respeito a ESBC: Reuso e Evolução. Em primeiro lugar devem existir componentes para o reuso, ou seja, deve haver um conjunto de componentes implementados e relacionados ao contexto desejados para serem descobertos, licenciados e facilmente utilizados. Além disso, deve haver também um modelo de componente que suporte a montagem e iteração dos componentes e um ambiente onde estes componentes possam existir e se comunicar. Para garantir o atendimento a esses requisitos é necessário a escolha de um modelo de componente que possua também uma infraestrutura de execução dos componentes implementados. Na próxima seção serão discutidos alguns modelos de componentes, em especial o modelo de componente selecionado para utilização neste trabalho.

2.3.1 Modelos de Componentes

Um modelo de componente define um conjunto de padrões para a implementação, nomenclatura, interoperabilidade, customização, evolução e implantação de componentes. Componentes de software necessitam de um modelo de referência que eles possam utilizar para fins de definição de interface, troca de mensagens e transferência de dados. Existem atualmente diversos modelos de componentes, tanto modelos com suporte comercial, como propostas acadêmicas, mas em todos os casos o modelo de componente especifica formas de o componente publicar sua interface, enviar mensagens e transferir dados (HOPKINS, 2000). Como exemplos de soluções comerciais podemos citar: COM/DCOM (*Distributed Component Object Model*) da Microsoft que, apesar de existirem adaptadores para outros modelos de componentes, são componentes essencialmente limitados para a plataforma Windows. O modelo CORBA, definido pelo *Object Management Group* (OMG), que é uma especificação independente de

linguagem de programação e plataforma. O modelo CORBA utiliza a linguagem IDL (*Interface Definition Language*) que possui padrões de interface para praticamente todas as plataformas usuais. O Enterprise JavaBeans ou especificação EJB da Sun Microsystems que oferece uma infraestrutura rica para a execução de componentes Java (SZYPERSKI; DOMINIK; STEPHAN, 1998). Em todos os três casos há o problema de dependências de plataforma, o que os torna inadequados para utilização neste trabalho, tendo em vista que uma das premissas básicas do nosso trabalho é que os componentes sejam independentes de plataforma.

Outros modelos de componentes foram analisados, que não possuem suporte comercial de grandes empresas de software ou são trabalhos acadêmicos. O objetivo era identificar um modelo de componente que fosse adequado para ser utilizado como base para a criação dos componentes propostos neste trabalho. Uma descrição breve dos modelos de componentes selecionados é feita a seguir.

O *OpenCOM* (COULSON et al., 2008) é um modelo de componente que oferece suporte para reconfiguração dinâmicas de sistemas, através do carregamento, retirada, vinculação e desvinculação de componentes em tempo de execução. Este modelo se encaixa em duas categorias de modelos de componentes: modelos do nível de aplicação e modelos de especificação de sistemas baseados em componentes. O modelo de componente denominado *K Component Model* (DOWLING; CAHILL, 2001) está inserido no ambiente de softwares dinâmicos, que tem a habilidade de modificar sua estrutura em tempo de execução, permitindo a extensão, customização e atualização dos serviços que proveem, sem a necessidade de recompilação ou reinicialização dos componentes. Os softwares dinâmicos são importantes no contexto de sistemas críticos que precisam operar em regime 24x7 ou sistemas que estão em constante adaptação ou dependentes das alterações no ambiente, a exemplo das aplicações móveis.

O *Fractal* (BRUNETON et al., 2004) é um modelo de componente genérico que tem por objetivo implementar, construir e gerenciar sistemas complexos, particularmente sistemas operacionais e middlewares. O principal diferencial deste modelo de componente é que ele possui características reflexivas, que é a habilidade que o programa tem de alterar sua estrutura e comportamento em tempo de execução. Desta forma, os componentes podem ser dotados de diferentes formas de controle. Os modelos analisados até esse ponto se propõem a resolver

um problema especifico, seja em análise preditiva de desempenho ou no suporte a evolução de sistemas sem a necessidade de interromper seu funcionamento.

O modelo de componente mais genérico avaliado foi o *Digital Content Component* (DCC) (SANTANCHÈ, 2006). O DCC é uma unidade de decomposição, compartilhamento, reuso e composição de conteúdo digital (SANTANCHÈ et al., 2009) Este modelo de componente foi criado com objetivo de encapsular qualquer tipo de conteúdo, desde código executável até objetos digitais complexos. Da mesma forma que um objeto digital complexo, um DCC agrega e encapsula um ou mais artefatos digitais e representa internamente as suas relações. Tal como os componentes de software, um DCC esconde detalhes internos da sua representação e expõe uma interface pública (SANTANCHE; MEDEIROS, 2004).

O DCC é definido a partir de um modelo abstrato composto de quatro subdivisões distintas: Conteúdo digital encapsulado, declaração de uma estrutura de gerenciamento que define como as partes dentro de um DCC estão relacionadas entre si, especificação das interfaces e metadados para descrever versão, funcionalidade, aplicabilidade e restrições de uso.

Existem dois tipos de DCC: os de processo e os passivos. Os DCCs de processo encapsulam software executável. Os componentes passivos encapsulam qualquer conteúdo digital não executável e não contém o software responsável por manipular este conteúdo (SANTANCHÈ et al., 2009). Tal manipulação é feita posteriormente a partir do acoplamento de um componente de conteúdo a um respectivo componente de processo que o manipula, chamado Companheiro.

O DCC é utilizado como modelo de componente em diversos trabalhos representando áreas de conhecimento diferentes, além de disponibilizar um framework para criação e execução de componentes na web.

Diante do ambiente diversificado de sistemas existente na Petrobras, especialmente na área de automação, o DCC se mostrou mais adequado com a possibilidade de reutilizar softwares legados através da especificação de componentes.

Um fato determinante para a adoção do DCC como modelo de componente neste trabalho foi a existência de ambientes de autoria implementados utilizando este componente. Como discutido anteriormente, permitir ao usuário construir suas próprias soluções é uma característica das soluções de automação utilizadas para construir monitores de processo na Petrobras. Na próxima seção discutiremos os

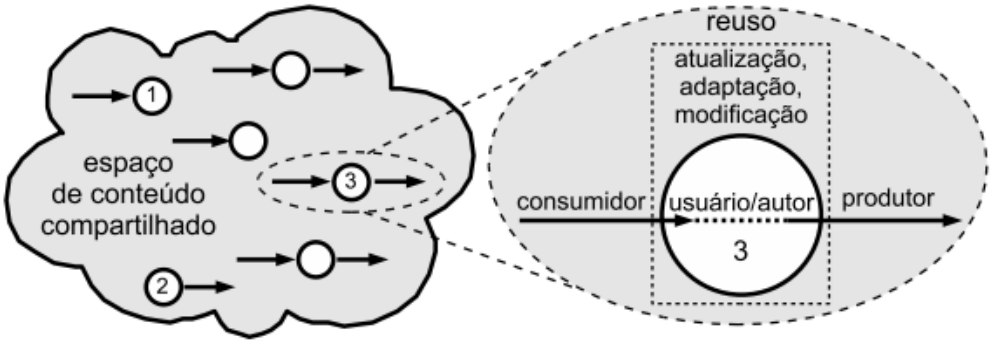
softwares de autoria e os trabalhos nessa área que utilizam o modelo de componente DCC.

2.4 AMBIENTE DE AUTORIA

Alguns sistemas de automação na Petrobras permitem que os usuários construam monitores de processo com as representações de equipamentos existentes em seu parque industrial. Os usuários utilizam estes softwares para construir e disponibilizar os monitores de processo sem a necessidade de possuir conhecimentos avançados em desenvolvimento de programas. O software de autoria é um programa que codifica o que o usuário quer realizar, sem que este necessite conhecer uma linguagem de programação, possibilitando através dele, criar outros programas seguindo fórmulas e receitas (BITTENCOURT et al., 2007; NUNES, 1997).

As principais aplicações dos softwares de autoria eram voltadas para os ambientes acadêmicos, na construção e aplicação de software educacional e com o foco nos usuários escolares: o professor e/ou o aluno. Os ambientes educacionais ainda são o principal foco das inciativas de pesquisa nessa área, porém tonar o usuário também o autor de suas soluções é uma funcionalidade que pode também ser explorada em outras áreas. No contexto de componentes de software, os ambientes de autoria são ferramentas que atuam como forma de potencializar o reuso e a evolução dos sistemas envolvidos.

Figura 6 – Definição do usuário-autor

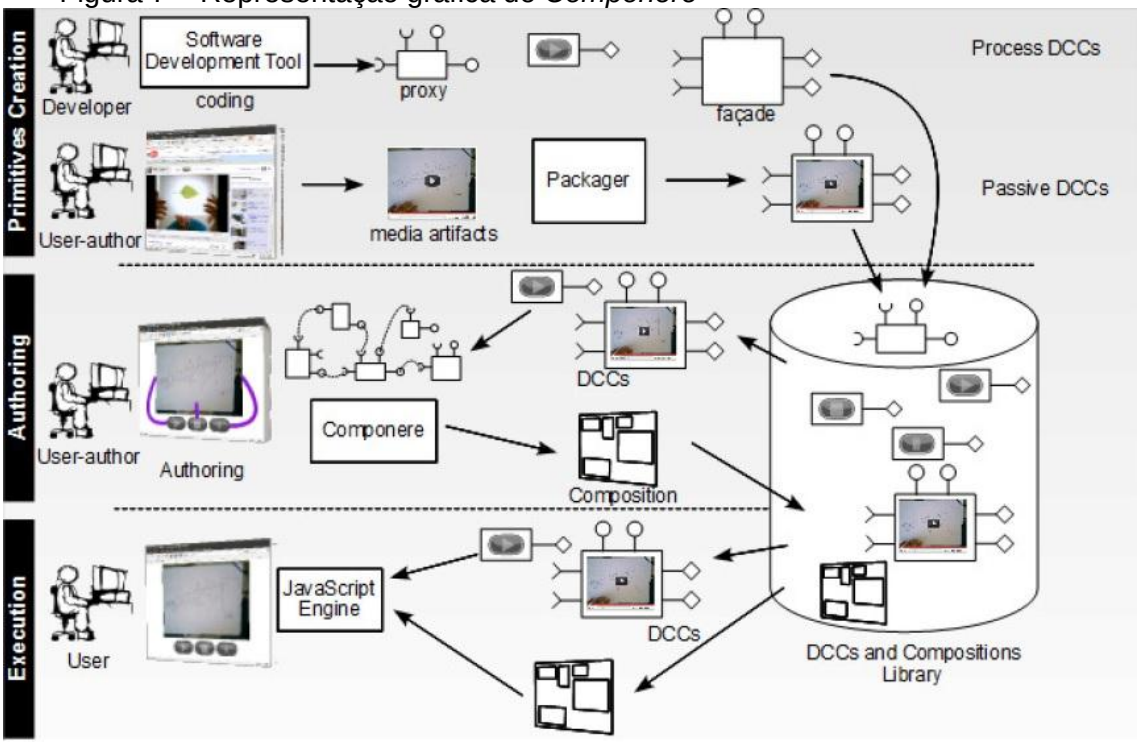


Fonte: Santanchè (2006).

O usuário-autor é definido em Santanchè (2006) como a combinação de dois papéis antes completamente segregados: o usuário (consumidor) e o autor (desenvolvedor) de produções multimídia e programas de computador. Com a evolução das ferramentas, dos padrões abertos e das redes de distribuição, o usuário-autor (Figura 5) surgiu como um novo tipo de usuário. Os usuários podem ser vistos como nós de um espaço de conteúdo compartilhado (Internet ou uma rede corporativa), consumindo conteúdo (setas de entrada) e reusando/produzindo conteúdo (setas de saída). Alguns dos nós/usuários são apenas consumidores (nó 1), enquanto outros são autores que produzem algo a partir do zero (nó 2). O usuário-autor alterna e combina os papéis de usuário e autor (nó 3)(SANTANCHÈ, 2006).

O *Componere* é um ambiente de autoria na Web baseado em componentes e completamente executado no navegador (SANTANCHÈ et al., 2009). Ele baseia-se no conceito de sistemas de autoria multimídia que permite aos autores criarem produtos multimídia utilizando uma combinação de outros produtos multimídia. O *Componere* oferece ao usuário-autor a possibilidade de criar suas próprias bibliotecas de componentes para domínios específicos ou aplicações. Algumas bibliotecas que já foram implementadas são referentes às áreas de biodiversidade, GIS e educação (SANTANCHE; MEDEIROS, 2004; SANTANCHÈ, 2006). O *Componere* foi construído com base no modelo de componente DCC.

Figura 7 – Representação gráfica do *Componere*



Fonte: Santanchè et al. (2009).

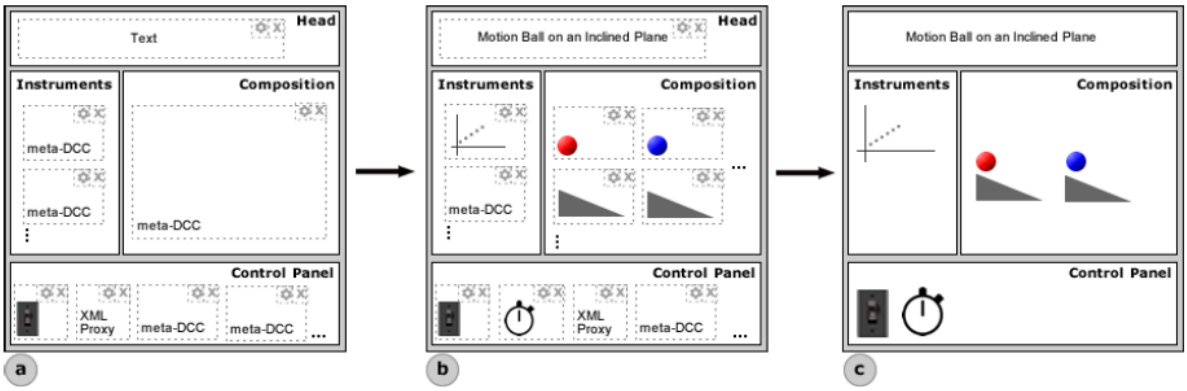
O processo de autoria do *Componere* é estruturado em três estágios principais: criação de elementos primitivos, autoria e execução (Figura 6). No primário estágio, os elementos primitivos são desenvolvidos como componentes e armazenados em uma biblioteca. Novos componentes são desenvolvidos por desenvolvedores de software ou pelo usuário final, dependendo do contexto e das habilidades de programação requeridas. Na fase de autoria, o usuário combina os elementos primitivos disponíveis para produzir composições. Os componentes são a matéria-prima para esta fase, que envolve a personalização e a composição dos componentes. O produto final desta fase pode também ser armazenado em uma biblioteca. Finalmente, na última etapa, as composições são executadas em um navegador.

O ambiente *Componere* foi utilizado como modelo para o desenvolvimento de outro ambiente de autoria baseado no DCC, o Metalaboratório, que introduz um conceito interessante no contexto de representação de processos, o Active Web Template e o componente meta-DCC.

O Metalaboratório é um ambiente na Web para produzir laboratórios de pesquisa científica baseados na composição de componentes, fornecendo ao

usuário uma perspectiva experimental (GOMES, 2014). Os usuários-autores podem, através desse ambiente, selecionar, customizar e combinar componentes, através de manipulação direta, tentando cenários alternativos. O Metalaboratório utiliza *Active Web* Templates, que é uma generalização de um documento Web para capturar e persistir os padrões de composição e o layout dos laboratórios montados pelo usuário-autor (GOMES, 2014). Nesse contexto, foi desenvolvido um componente denominado Meta-DCC. Ele funciona como uma marcação realizada no *template* de laboratório criado, e essa marcação permite adicionar em seu lugar, em tempo de execução, um novo componente do tipo instrumento. Ao executar o *template* o usuário pode selecionar um componente de uma biblioteca e inserir este componente no lugar do Meta-DCC.

Figura 8 – Template do Metalaboratório



Fonte: Gomes, Hermano e Santanchè (2014).

A Figura 8 ilustra a abordagem em três passos do Metalaboratório para produzir laboratórios baseados em Template Ativos, uma denominação utilizada em (GOMES, 2014) para identificar *templates* que são executáveis e autoconfiguráveis, em contraste aos *templates* passivos utilizados em outras abordagens. O Passo (a) mostra a estrutura inicial do Active Web Template. Um usuário designer de *templates* constrói um *template*, geralmente buscando uma composição e generalização de laboratório pré-existente e substituindo os componentes por Meta-DCCs. No passo (b) um autor instancia um *Active Web Template*. O autor personaliza o referido *template* através dos Meta-DCCs, que de acordo com a sua especialidade, permitem aos autores: selecionar a fontes de componentes, arrastar e posicionar os componentes em um espaço de tela, dentre outras funções. Na

última etapa (c), o ambiente de execução converte o modelo em um documento final Web. O modelo de Meta-DCCs fornece instruções para o ambiente de execução dos DCCs para que estes sejam inseridos e adaptados (GOMES, 2014).

A partir do modelo de captura da composição e layouts adotados no Metalaboratório e utilizando a mesma abordagem de criação de planta do ProcessBook, foi criado um conjunto de componentes para permitir a criação de monitores de processo. Nos capítulos três e quatro discutiremos a família de componentes proposta para modelar os equipamentos e a família de componentes proposta para permitir o desenho de um processo de produção de petróleo e gás.

O objetivo da família de componentes de Processo é a representação do processo de produção de petróleo em instalações terrestres. Nesse contexto é necessário posicionar o modelo de representação do processo escolhido, para o componente, dentro do universo de Gerenciamento de Processos de Negócios (BPM). Na próxima seção discutiremos o conceito de processo de negócio com o foco na notação utilizada pela família de componentes de Processo para representar o processo de produção de petróleo e gás.

2.5 PROCESSO DE NEGÓCIO

O conceito de processo utilizado neste trabalho está relacionado com a representação do processo de produção de petróleo em instalações terrestres, o qual é definido pela disposição de equipamentos e as suas relações. Uma consideração importante que deve ser feita é que a família de componentes proposta não tem o objetivo de modelar do processo de negócio, de acordo com as premissas do BPM. Ele tem o único propósito de documentar o processo existente e pode ser utilizado para cobrir a primeira etapa de modelagem de processos, que aborda o desenho do processo atual usando uma notação de fácil compreensão para todos os envolvidos. O termo modelagem de processo é utilizado neste trabalho para facilitar a compreensão, mas os componentes definidos não abordam qualquer atividade definidas em BPM, mas pode ser usado como uma ferramenta que suporta a fase de concepção do processo no seu estado atual. A família de componentes de processo definidos para o ambiente de autoria proposto, que será apresentada nos próximos capítulos, segue a notação padrão para o processo de desenho, BPMN. Nas seções abaixo é feita uma introdução ao conceito de processo

de negócio, ao conceito de BPM e o padrão de notação para o desenho de processo denominado BPMN, que é uma notação utilizada pelo componente de *Processo* proposto neste trabalho.

2.5.1 Conceito de Processo de Negócio

O processo de negócio, assim como o processo produtivo, é composto por diversas etapas de produção ou atividades a serem executadas (DE SORDI, 2008). A ideia de dividir o trabalho em atividades sequenciais se constitui no único ponto de consenso entre alguns dos principais pesquisadores que definiram o que é o processo de negócio (DE SORDI, 2008). As organizações precisam de meios para lidar com as mudanças inesperadas no negócio, atender às expectativas de clientes em constante mudança e também implementar novos padrões tecnológicos (NETTO, 2006). Alcançar tudo isso, segundo Smith e Fingar (2002), requer uma linguagem de processo padrão que permita aos parceiros a participação no desenho dos processos e o entendimento mútuo detalhado das operações. Atualmente o Gerenciamento de BPM é fundamentalmente voltada ao gerenciamento dos processos (SMITH; FINGAR, 2002). O BPM é uma síntese entre a representação de processos e as tecnologias de colaboração, que promove a remoção dos obstáculos que bloqueiam a execução dos objetivos organizacionais, ou seja, BPM é, para os autores, a convergência das teorias da administração com tecnologias modernas – desenvolvimento de aplicativos, integração de sistemas de informação, arquitetura orientada ao serviço, workflow, XML e Web Services (NETTO, 2006).

Tendo em vista os conceitos apresentados sobre gerenciamento de processos de negócio, o sistema WebPlantas proposto neste trabalho pode ser utilizado como uma ferramenta que fornece subsídios para o usuário na primeira fase do BPM, que trata do desenho do estado atual do processo que se deseja controlar. Porém o escopo de atuação do WebPlantas é especializado no processo de produção de petróleo em instalações terrestres. A família de componentes de processo definidas para o WebPlantas segue o padrão de notação para o desenho de processos definido para o BPM discutido a seguir.

A organização Business Process Management Initiative (BPMI) desenvolveu um padrão de notação para o desenho de processos denominado Notação de Modelagem de Processos de Negócio (Business Process Model and Notation -

BPMN). O objetivo principal deste padrão é prover uma notação que seja prontamente entendível por todos os usuários do negócio, desde o analista de negócio que cria os primeiros desenhos dos processos até os desenvolvedores técnicos responsáveis por implementar a tecnologia que irá executar esse processo, e finalmente até as pessoas do negócio que irão gerenciar e monitorar os processos (WHITE, 2004).

2.5.2 Padrões de Notação

O BPMN define o Diagrama de Processo de Negócio (BPD), que é baseado em uma técnica de fluxograma adaptada para a criação de modelos gráficos de operações de processos de negócios (WHITE, 2004). Um modelo de processo de negócio, então, é uma rede de objetos gráficos, que representam atividades (ou seja, trabalho) e os controles de fluxo que definem a ordem de desempenho. O BPD é construído por um conjunto de elementos gráficos, que permitem facilmente desenvolver digramas simples que serão reconhecidos pela maioria dos analistas de negócio (WHITE, 2004). As quatro principais categorias de elementos gráficos são: Objetos de Fluxo, objetos de conexão, raias e artefatos. Para o nível de detalhamento do processo representando pelos componentes de processo propostos neste trabalho, são relevantes apenas alguns elementos das categorias definidas pelo BPMN. Esses elementos são detalhados abaixo.

O elemento Evento é representado por círculos. É algo que acontece durante o curso do processo de negócio. Esses eventos afetam o fluxo do processo e geralmente tem uma causa ou um impacto. Existem três tipos de eventos, baseados na maneira que eles afetam o fluxo: início, imediatamente e fim. Na  temos a representação de um evento início: círculo com linhas simples, uma representação de um evento imediato: círculo com linha dupla e uma representação de um evento fim: círculo com linha grossa em negrito.

Figura 9 – Representação de eventos em BPMN

Fonte White (2004).

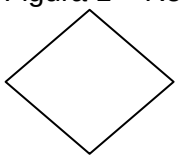
O elemento Atividade é representado por um retângulo com pontas arredondadas, como na  . É um termo genérico para o trabalho que a companhia executa, podem ser atividades ou sub-processos.

Figura 10 – Representação de uma atividade em BPMN

Fonte: White (2004).

O elemento Gateway é representado por um formato de losango, como mostrado na Figura 11. É utilizado para controlar a convergência e a divergência em um fluxo sequencial.

Figura 1 – Representação de *gateway* em BPMN



Fonte: WHITE (2004).

O elemento Fluxo de sequência é representado por uma linha sólida com uma cabeça de seta sólida, como mostra a Figura 5.4. É utilizada para mostrar a ordem, a sequência, que as atividades serão executadas no processo.

Figura 12 – Representação de um fluxo de sequência em BPMN

Fonte: White (2004).

O elemento Grupo é representado por um retângulo com bordas arredondadas desenhado com linhas pontilhadas, como mostrado na Figura 4.5. O grupo pode ser utilizado com o propósito de documentação ou análise, mas não afeta a sequência do fluxo.

Figura 13 – Representação de um Grupo em BPMN

Fonte: White (2004).

2.6 CONSIDERAÇÕES FINAIS

A ferramenta proposta nesse trabalho tem o objetivo de permitir ao usuário executar as atividades de criação de plantas realizadas na aplicação *ProcessBook*, ampliando as funcionalidades e facilitando a integração com outras ferramentas e fontes de dados. Para alcançar este objetivo é necessário conhecer as fontes primárias de dados na área de automação, detalhando as camadas definidas para as aplicações de automação e as fontes de dados primárias das variáveis do processo, o que foi realizado neste capítulo.

As aplicações na área de automação são fundamentais para o controle e monitoramento do processo nas indústrias, porém, as empresas fabricantes oferecem softwares fechados, inflexíveis e difícil integração com as demais aplicações existentes nas empresas que consomem esses sistemas. A criação de uma aplicação aberta e flexível pode contribuir para o trabalho dos diversos interessados nessa área, especialmente, os engenheiros de processo e operadores da indústria de petróleo e gás.

O ambiente de composição proposto neste trabalho tem o objetivo de ser um consumidor das informações geradas nos sistemas PIMS e demais repositórios através de componentes de software. O referido ambiente permite a criação de plantas para monitoramento baseada em componentes de software, o que a configura como uma solução aberta para desenvolvimento colaborativo e flexível para integração com qualquer sistema ou fontes de dados de interesse.

3 COMPONENTES DE SOFTWARE PARA REPRESENTAR EQUIPAMENTOS UTILIZADOS NA PRODUÇÃO DE PETRÓLEO E GÁS TERRESTRE

A construção de sistemas através da composição de componentes pressupõe a existência de uma infraestrutura mínima para que os componentes possam ser reutilizados para a produção de novos sistemas. Essa infraestrutura mínima compreende o modelo de componente, um conjunto mínimo de componentes para serem descobertos, licenciados e facilmente utilizados e um ambiente de execução que suporte a montagem e iteração dos componentes, isto é, um ambiente onde estes componentes possam existir e se comunicar (HOPKINS, 2000).

Como já discutido anteriormente, o modelo DCC, possui um framework implementado e disponível para utilização que trata da criação, execução e iteração entre os componentes na Web. O framework *Componere* (HERMANO; SANTANCHÈ, 2015) permite instanciar componentes na Web e provê toda a infraestrutura de troca de mensagens entre componentes além de dar suporte a herança de componentes. Desta forma, a infraestrutura disponibilizada pelo DCC e Componere contemplam duas premissas fundamentais para a construção de plantas industrias através da composição de componentes na Web. A terceira premissa, isto é, o repositório de componentes relacionados com o domínio da aplicação foi desenvolvido neste trabalho. O repositório é dividido em duas grandes classes de acordo com contexto de utilização: Uma família de componentes denominada de Equipamento e outra família de componentes denominada Processo. No restante desse capítulo discutiremos a estrutura proposta para o repositório e os componentes da família de Equipamento e no próximo capítulo discutiremos os componentes da família de Processo.

3.1 ESTRUTURAÇÃO DE UM REPOSITÓRIO DE COMPONENTES

O desenvolvimento de sistemas utilizando a Engenharia de Software Baseado em Componentes requer a existência de um repositório onde os componentes possam ser descobertos e utilizados. Diversos aspectos estão envolvidos na estruturação de um repositório de componentes. O Repositório deve possuir estruturas que suportem a descoberta de funcionalidades disponibilizadas pelos

componentes. É importante, também, que os componentes estejam bem organizados no repositório, através de uma especificação de domínios de atuação dos componentes. Outro aspecto importante em um repositório é a necessidade de possuir uma estratégia para tratar da integração de repositórios distribuídos (ALVES, 2011).

O repositório proposto neste trabalho é classificado em duas grandes famílias de componentes de acordo com o domínio de atuação, a família de componentes de Equipamento, que representam os equipamentos utilizados no processo de produção de petróleo e gás e uma família de componentes de Processo, que tratam das relações de composições entre os componentes da família de Equipamentos como forma de representar o processo de produção das instalações de petróleo e gás.

A família de componentes Equipamentos possui uma classificação em função do contexto da utilização do equipamento no processo de produção. Essa classificação é importante também para a família de equipamentos de Processo, pois permite a definição de restrições semânticas na construção de modelos de instalação de produção de petróleo. A seguir apresentamos uma classificação dos equipamentos utilizados na produção de petróleo e gás que foi utilizada na definição dos componentes que fazem parte da família de componentes Equipamento.

3.1.1 CLASSIFICAÇÃO DE EQUIPAMENTOS

Os componentes desenvolvidos neste trabalho representam um grupo de instrumentos de acordo com uma classificação proposta neste trabalho. A classificação dos equipamentos contempla quatro grandes grupos: Produção, Medição e Armazenamento, Mediação Auxiliar e Transporte. Assim, todos os componentes da família *Equipamento* se enquadram em uma dessas classes, de acordo com as características do equipamento e na fase do processo que ele é utilizado. Especializações do componente *Equipamento* são o equivalente virtual aos instrumentos existentes fisicamente. A classificação foi criada com base na descrição do processo de produção feita em (RIBEIRO, 2003) e nos modelos de dados dos sistemas operacionais utilizados na Petrobras. A seguir uma breve descrição da classificação proposta.

a) Produção

Nesse grupo estão os instrumentos associados ao início do processo de produção de petróleo e gás e seus processos elementares: desidratação, tratamento e separação do gás (FUNDAMENTAIS, 2013; RIBEIRO, 2003). Exemplo: poços, separadores de duas fases, separadores de três fases, vasos para tratamento.

b) Medição e Armazenamento

Nesse grupo encontram-se os equipamentos que representam o núcleo do processo de produção de Petróleo e Gás, pois são utilizados para armazenar os fluidos de interesse e também realizar a medição para contabilização com fins financeiros e legais. Estes são, portanto, os equipamentos mais fiscalizados e controlados pelas empresas produtoras de petróleo e pelos órgãos controladores governamentais.

Após sair dos poços e passar pelos primeiros separadores de fluidos, o petróleo deve satisfazer determinadas especificações para poder entrar no sistema de transportes. Assim, o petróleo tratado aguarda a transferência de custódia em um ou mais tanques de armazenamento nas áreas de produção. Como exemplos deste elemento podem citar: Tanque e Medidores de Vazão.

c) Medição Auxiliar

Nesse grupo estão os instrumentos utilizados para a medição, alarme, monitoração e controle das variáveis do processo industrial. As variáveis típicas incluem, mas não se limitam a pressão, temperatura, vazão, nível e análise. Como exemplo tem-se: termômetros, analisador de pressão e analisador de BSW.

d) Transporte

Existem diversos tipos de instalações no processo de produção de petróleo e esse óleo pode circular entres diversas instalações até atingir seu destino final a depender da configuração do processo da empresa produtora. Existem, por exemplo, instalações que não fazem nenhum tipo de separação ou tratamento do óleo, apenas o retiram de um poço e armazena em um

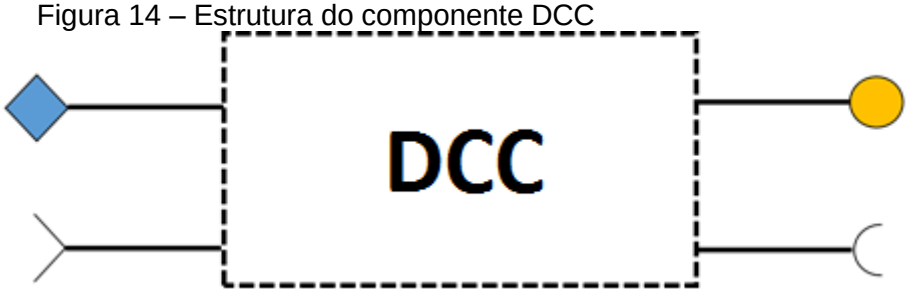
tanque, sendo esse o tipo mais simples de estação chamada de ponto de coleta. De tempos em tempos um caminhão do tipo carreta é enviado ao ponto de coleta para coletar a produção e levá-la para uma estação de tratamento para processamento desse fluído. Esse processo é dado o nome de transferência de petróleo, que é o meio utilizado para realizar a movimentação entre as estações e a transferência para o refino que representa o fim do processo de produção. Como exemplo de Equipamento do tipo Transporte podemos citar as carretas e os dutos.

3.1.2 ESTRUTURA DO DCC

O ponto de partida para a implementação dos componentes é a definição da estrutura básica do componente que servirá de molde para o desenvolvimento. A utilização do ambiente webframework *Componere,* disponibilizado em (HERMANO; SANTANCHÈ, 2015) e a estrutura do componente DCC definida em (GOMES, 2014) foi utilizada neste trabalho para definir a estrutura básica utilizada em todos os componentes proposto.

A estrutura utilizada na implementação dos componentes Equipamento utilizou uma estrutura similar a estrutura do DCC utilizada no Metalaboratório (GOMES, 2014), exibida.na Figura 14.

Figura 14 – Estrutura do componente DCC



Fonte: Autoria própria do autor desta dissertação (2015).

Na Figura 14 é mostrada a estrutura utilizada na implementação do componente DCC Equipamento. A caixa pontilhada no centro da figura representa um DCC, as setas com o círculo e o semicírculo nas pontas representam as interfaces providas e requeridas, respectivamente. O componente que implementa uma interface requerida, solicita o serviço do componente que está conectado a ele.

O componente que implementa uma interface provida executa o serviço requerido e entrega o resultado ao solicitante. Ainda na Figura 14, a seta com o diamante na ponta e seta com meio diamante na ponta representam as interfaces *Publish/Subscribe*, respectivamente. Componentes que implementam a interface *Publish* enviam mensagens, aos inscritos quando determinado evento ocorre. Os componentes que implementam a interface *Subscribe* são os inscritos. Estes componentes registram interesse em um determinado evento e irão receber mensagens desse evento apenas pelo seu método de notificação.

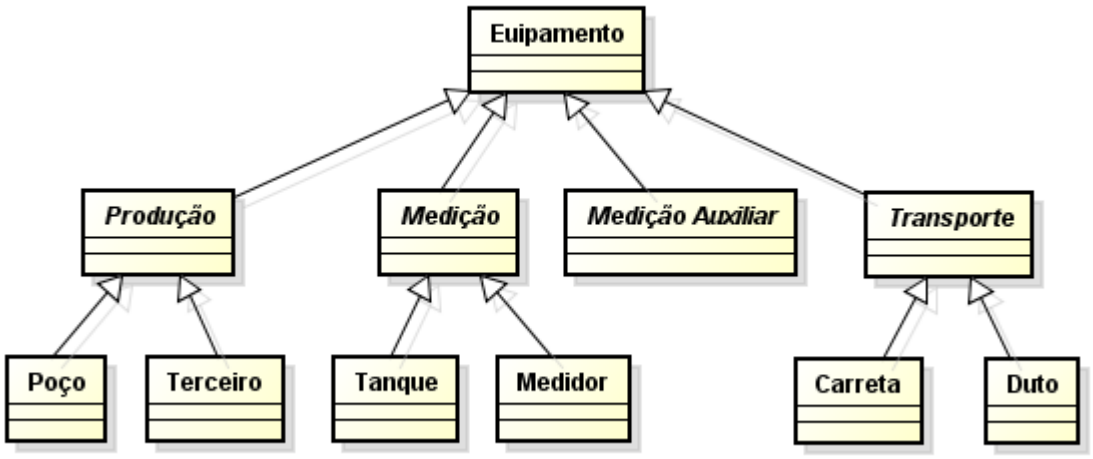
3.2 FAMILIA DE COMPONENTES EQUIPAMENTO

Nas aplicações de automação que permitem a criação de plantas de processo, os equipamentos são representados por imagens que são associadas a uma *Tag*, Uma Tag é equivalente a um termo ou uma palavra chave que identifica unicamente o equipamento no repositório de dados de automação. Baseado na representação visual e da associação com uma Tag são criadas telas de monitoramento das instalações.

No ambiente proposto neste trabalho o usuário tem a liberdade de escolher o repositório de onde os dados serão recuperados e poderá escolher a forma como esse equipamento será representado graficamente. Os componentes propostos podem ter relação direta com os equipamentos da planta, a exemplo do componente DCC *Equipamento* (Figura 15), que define unicamente um instrumento utilizado no processo.

O DCC Equipamento representa os diversos equipamentos existentes em uma planta de produção de petróleo e gás. Os DCCs são organizados em classes OWL de acordo com sua funcionalidade. Cada classe DCC define a semântica de um componente em uma taxonomia e suas interfaces providas/requeridas. Uma amostra do conjunto de equipamentos que podem ser representados utilizando este componente é mostrado na Figura 15. Utilizamos a notação UML para mostrar relações entre classes OWL como forma de facilitar o desenho e o entendimento.

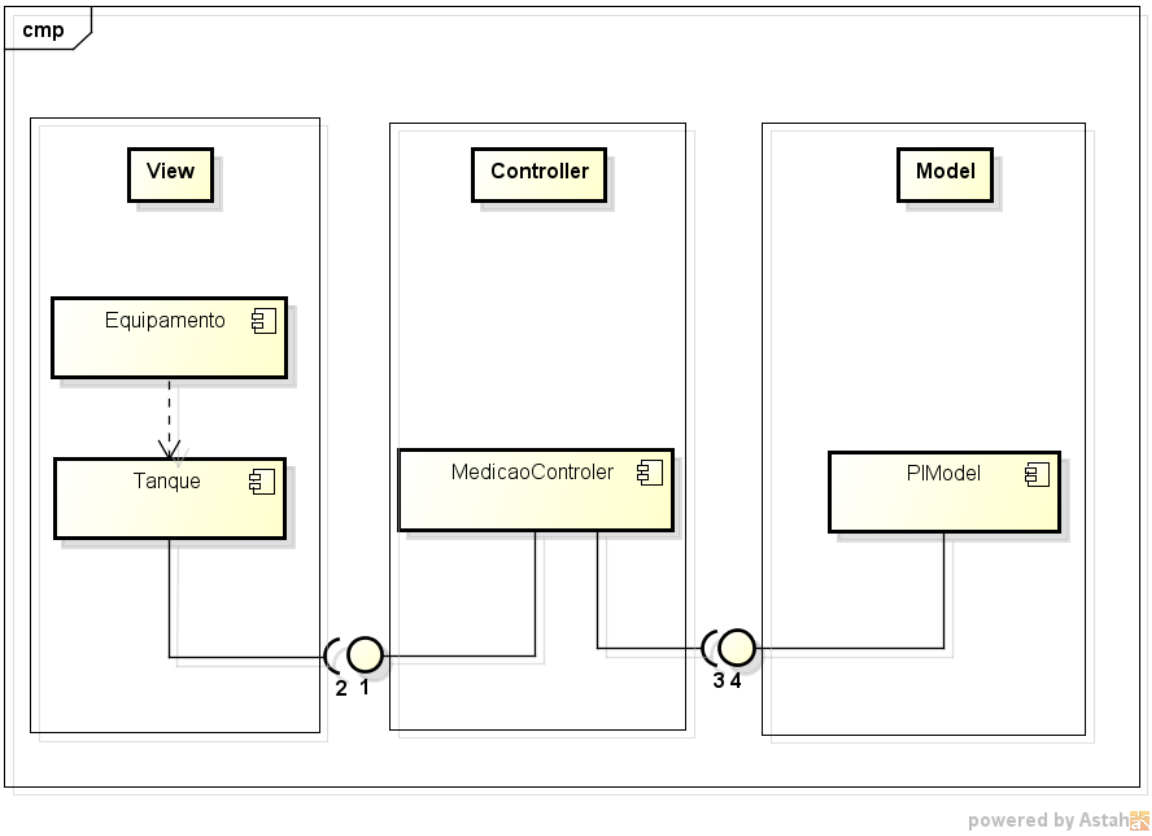
Figura 15 – Equipamentos representados pelo DCC *Equipamento*



Fonte: Autoria própria do autor desta dissertação (2015).

Na Figura 16 temos o conjunto de componentes que formam a família de componentes para representar equipamentos proposta. Ela é sistematizada em torno de um conjunto de classes OWL, divididos em dois grupos. O grupo 1 está relacionado com a classificação do equipamento em função da sua utilização no processo de produção. O grupo 2 está relacionada com o perfil do componente de acordo com o padrão *Model-View-Controller* (MVC), áreas destacadas na Figura 16. Os componentes *Model* e *Controller* realizam um trabalho acessório, isto é, estes componentes são utilizados para recuperar informações dos repositórios e atualizar valores de atributos de acordo com parâmetros estabelecidos pelo usuário, respectivamente.

Figura 16 – Relacionamento entre Componentes Equipamento



Fonte: Autoria própria do autor desta dissertação (2015).

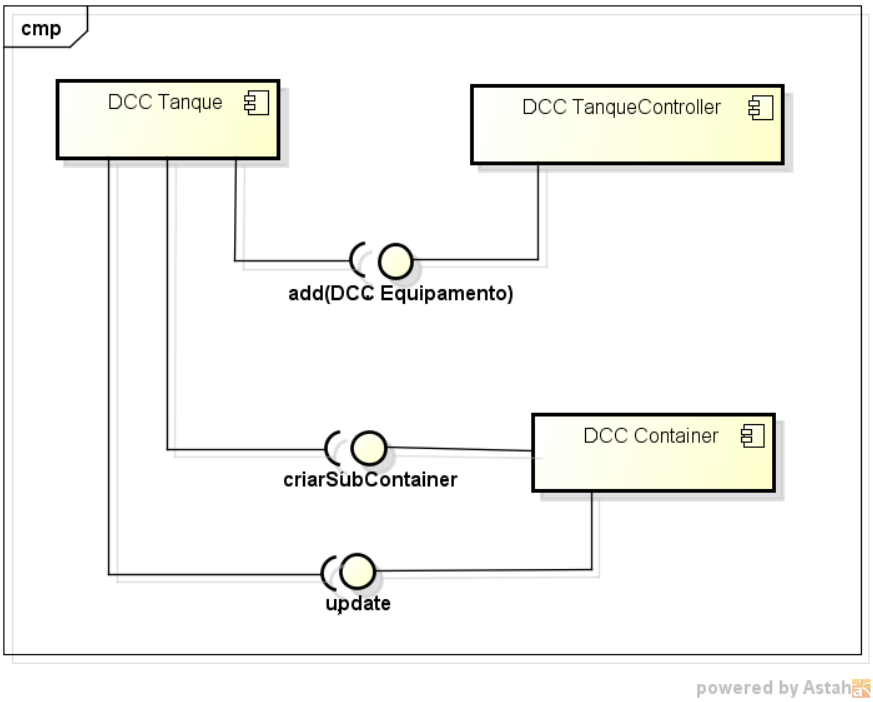
Um DCC *Equipamento* tipicamente especializa uma classe de cada um dos grupos destacadas na Figura 16. Um componente que pertence a uma classe significa que ele atende a um conjunto de interfaces relacionadas com o papel do componente e a maneira como ele se relaciona com o ambiente e com os demais componentes. O modelo de equipamento possui classes OWL relacionadas com o papel do componente no ambiente proposto seguindo o padrão MVC (Figura 16). O padrão MVC tem como objetivo a separação da entrada do usuário, acesso a dados e regras de negócio em três camadas: *model, view* (interface) e *controller*. O *model* representa componentes que tratam com diferentes fontes de dados, a exemplo de bancos de dados e sensores. *View* gerencia entradas e saídas do usuário através da interface da aplicação, e é responsável pela representação visual dos equipamentos e dos valores de suas variáveis. O *controller* intermedia a relação entre *model* e *view* e controla todo o processo.

3.2.1 DCC Equipamento

Para ilustrar a criação e implementação dos componentes propostos foram utilizados os equipamentos existentes em uma instalação do tipo Ponto de Coleta. Este tipo de instalação é uma das instalações mais simples no processo de produção de petróleo e gás. Em um Ponto de Coleta existem pelo menos os seguintes equipamentos: Poço, Tanque e Carreta. Para discutir a implementação de um DCC Equipamento foi escolhido o equipamento Tanque.

O Componente *DCC Equipamento* que representa um Tanque, ou simplesmente *DCC Tanque,* tem as atribuições de definir a representação gráfica do equipamento na tela do navegador e definir os atributos que serão monitoradas pelo aplicativo através de um componente *Controller*. Na Figura 17 é possível ver um trecho de código da criação do *DCC Tanque*, utilizando o framework *Componere*. No trecho destacado como (A) da Figura 17 temos a inicialização do componente Web e das variáveis utilizadas no código para armazenar os tanques. Um componente *DCC Tanque* pode criar e controlar mais de um tanque. O trecho destacado como (B) mostra a definição das interfaces utilizadas pelo componente, neste caso é consumida uma interface disponibilizada pelo componente *Container*, para o desenho do equipamento na tela do navegador, e consumida uma interface disponibilizada pelo componente *Controller* para criação de um painel onde as variáveis de interesse serão visualizadas. O trecho destacado como (C) mostra a utilização de um objeto JSON (ECMA INTERNATIONAL, 2013) para criar uma representação de um tanque existente na planta mapeada. São definidos os atributos que definem unicamente o tanque nos diversos repositórios, bem como as variáveis de interesse que devem ser observadas no monitor da planta que será construído. Nesse exemplo são definidas a TAG do tanque como forma de representa-lo unicamente e o atributo volume que representa o estoque atual de petróleo existente no tanque.

Figura 17 – Relacionamentos do DCC Tanque



Fonte: Autoria própria do autor desta dissertação (2015).

Figura 18 – Implementação de um DCC Equipamento Tanque

A

```
var tanque;
var tanques=[];
tanque = new Component('./tanque.js');
tanque.extend('./ui-element.js');
```

B

```
tanque.require('tqControleInterface', function (tqControler,done) {
  this.tqControler = tqControler;
  done();
});

tanque.require('containerInterface', function (container,done) {
  this.container = container;
  done();
});
```

C

```
tanque.install(function (done) {
tanque.start(function (done) {
  this.create();

    var equipamento1 = JSON.stringify({
        tipo    : 'Medicao',
        tag     : 'TQ001',
        var_1   : 'volume',
        var_2 : 'temperatura',
        val_1:  '',
        val_2:  '',
        imagem: './img/tq4.jpg'
    });

    tanques.push(equipamento1);
```

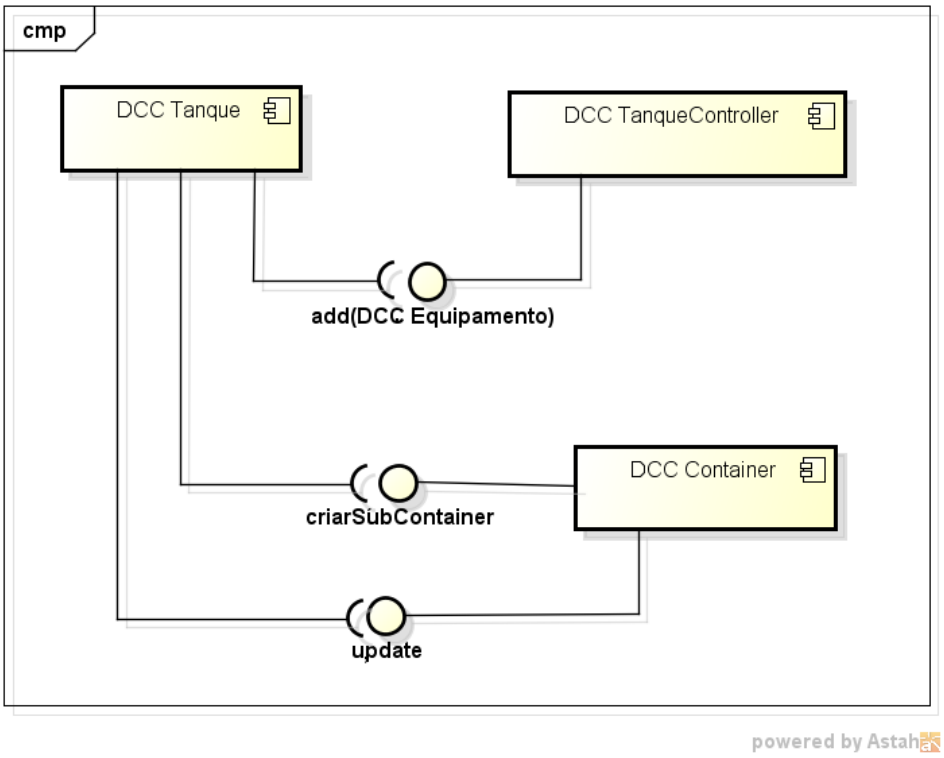
Fonte: Autoria própria do autor desta dissertação (2015).

3.2.2 DCC Controller

Toda informação necessária para monitorar o processo de produção de petróleo é representada pelo componente que especializa a classe *Controller*. Este componente trata dados obtidos de diversos repositórios e pode ser configurado pelo usuário para atualizar os valores monitorados em um determinado intervalo de tempo. A informação é obtida de fontes diferentes e com diferentes significados. A informação sobre o status de um equipamento, por exemplo, poderá ser obtida através de uma consulta SQL, uma instrução de automação ou através da obtenção de valor historiado por um software PIMS.

Cada componente *Controller*, definido pelo usuário, pode implementar um conjunto características do componente Equipamento com o qual se relaciona, como por exemplo, a indicação visual do volume existente dentro de um tanque e a criação e a manutenção de um quadro com os valores das variáveis definidas para o equipamento atualizadas em tempo real. Na Figura 18 temos um componente *Controller*, fazendo interface com um repositório PI mapeada por um componente Model.

Figura 19 – Relacionamentos do DCC Controller



Fonte: Autoria própria do autor desta dissertação (2015).

Figura 20 – Implementação de um DCC *Controller*

A

```
var tqControle;
var equipamentosM=[];
tqControle = new Component('./TqController.js');

tqControle.extend('./ui-element.js');
```

B

```
tqControle.listen('timeUpdate', function () {
  this.get();
});

tqControle.provide('tqControleInterface', function (done) {
  done({'add' : this.add, 'criaControle' : this.criaControle});
});

tqControle.require('PIInterface', function (PIModel,done) {
  this.PIModel = PIModel;
  done();
});
```

C

```
tqControle.install(function (done) {

  this.criaControle = function (obj,str){

    equipamento = JSON.parse(obj);
    $("<div class='controle'  id='" + 'c' + equipamento.tag + "'></div>").appendTo("#"+str);
    $("#c"+equipamento.tag).append(equipamento.var_1+':');
    $("#c"+equipamento.tag).append('<br>');
    $("#c"+equipamento.tag).append('<span id=\"'+equipamento.tag+'_'+equipamento.var_1+'\"></span>');
    $("#c"+equipamento.tag).append('<br>');
    $("#c"+equipamento.tag).append(equipamento.var_2+':');
    $("#c"+equipamento.tag).append('<br>');
    $("#c"+equipamento.tag).append('<span id=\"'+equipamento.tag+'_'+equipamento.var_2+'\"></span>');
```

Fonte: Autoria própria do autor desta dissertação (2015).

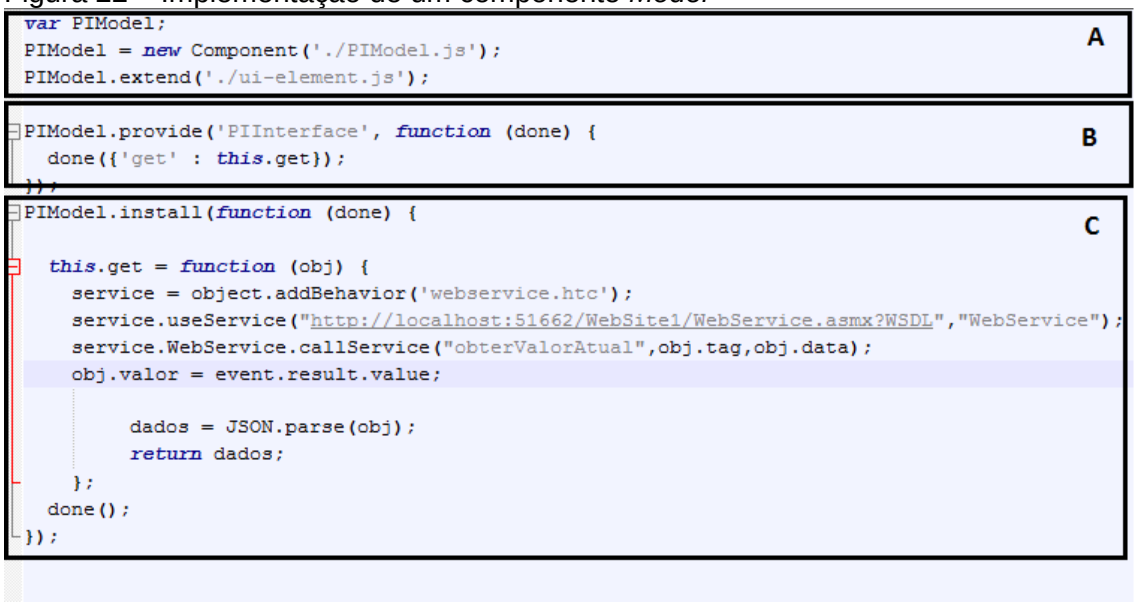
Parte do código referente a implementação de uma instância do componente *Controller*, denominado *TQController*, utilizando o framework *Componere* é mostrado na Figura 18. O código foi divido em três áreas. Na área A é exibido o trecho de código onde são definidas as variáveis utilizadas e um componente do tipo Controle é instanciado utilizando a sintaxe do framework *Componere*. Na área B as interfaces providas e requeridas são declaradas. Neste trecho pode-se observar que o componente é configurado para escutar um evento, denominado *timeUpdate*. Assim, todo cada vez que o evento é disparado o método *get* do *TQController* é executado. As interfaces providas são disponibilizadas através dos métodos *add* e *criaControle.* Finalmente, na área C temos o ciclo de instalação (*install*) do framework *Componere*. Nesse ciclo, os métodos declarados na interface são chamados e no caso do *TQController* a interface disponibilizada *criaControle,* cria um elemento HTML onde são exibidos na tela os atributos definidos pelo componente Equipamento. Todos os componentes são atualizados de acordo com o evento *timeUpdate.*

3.3 DCC MODEL

Alguns componentes *Controller* podem ser relacionados com um componente denominado *Model*. O componente *Model* é responsável por estabelecer conexões com os repositórios ou sistemas supervisórios através de protocolos de comunicação, deixando os detalhes de cada fonte de dados transparente para o usuário final. Um componente *Model* irá prover uma interface para cada fonte de dados disponível. Por exemplo, um *Controller* pode ter uma interface requerida com o repositório PI que é provida por uma interface de um componente *Model* conectado a uma base de dados PI.

A implementação do componente PIModel (Figura 19), disponibiliza uma interface que é consumida pelo componente *TQController* discutido na seção anterior. O método disponibilizado consome um Webservice que retorna dados obtidos da base de dados historiada do sistema PI. Partes do código referentes a implementação do *PIModel* utilizando o framework *Componere* é mostrado na Figura 19. No trecho A da figura são definidas as variáveis utilizadas e o componente do tipo *Model* é instanciado utilizando a sintaxe do framework *Componere*. No trecho B a interface provida *PIInterface* é declarada através do método *get*. Finalmente no trecho C da figura temos mais uma vez o denominado ciclo de instação do framework *Componere*, onde os métodos declarados na interface são implementados. No caso do *PIModel*, o método disponibilizado pela interface implementa a chamada do serviço Web *obterValorAtual()*, que recebe como entrada um identificador (Tag) PI e uma data/hora e retorna um valor numéric. Este serviço foi desenvolvido na linguagem.NET para ser utilizado neste trabalho.

Figura 22 – Implementação de um componente *Model*




```
var PIModel;
PIModel = new Component('./PIModel.js');
PIModel.extend('./ui-element.js');
```

```
PIModel.provide('PIInterface', function (done) {
  done({'get' : this.get});
});
```

```
PIModel.install(function (done) {

  this.get = function (obj) {
    service = object.addBehavior('webservice.htc');
    service.useService("http://localhost:51662/WebSite1/WebService.asmx?WSDL","WebService");
    service.WebService.callService("obterValorAtual",obj.tag,obj.data);
    obj.valor = event.result.value;

        dados = JSON.parse(obj);
        return dados;
    };
  done();
});
```

Fonte: Autoria própria do autor desta dissertação (2015).

3.4 CONSIDERAÇÕES FINAIS

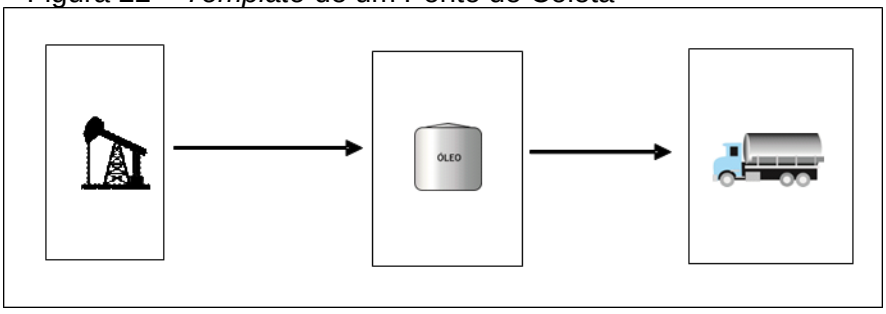
Neste capítulo foram definidos os componentes que representam os equipamentos de uma planta de produção de petróleo e gás com base em uma classificação proposta e de acordo com a etapa do processo onde eles serão utilizados. A classificação é importante, pois possibilita a criação de contextos semânticos para a utilização dos equipamentos. Os componentes propostos modelam também as interfaces com os diversos repositórios. O componente *DCC Equipamento* representa os diversos equipamentos utilizados em uma planta de produção de óleo e gás, ele utiliza o componente *DCC Controller* para recuperar dados dos atributos mapeados de um repositório. Os repositórios são representados por componentes *DCC Model*.

A escolha do modelo de componentes DCC como base para os componentes de equipamento se mostrou bastante adequada, pois o DCC além de possuir um framework para suporte ao desenvolvimento implementação dos componentes, utiliza a linguagem Javascript, o que permite a execução do ambiente em navegadores Web (HERMANO; SANTANCHÈ, 2015).

4 COMPONENTES DE SOFTWARE PARA REPRESENTAR O PROCESSO DE PRODUÇÃO DE PETRÓLEO E GÁS TERRESTRE

Um monitor de processo para controlar a produção de petróleo de uma planta terrestre pode ser construído através da composição dos componentes mais simples, que representam os equipamentos utilizados na planta. Porém, na Petrobras observa-se que as instalações de produção de petróleo seguem padrões rígidos de organização dos equipamentos utilizados nas instalações. Estes padrões definem etapas do processo que representam os diversos tipos de instalação de produção de petróleo. Cada tipo de instalação define um arranjo dos equipamentos, que consequentemente influencia na complexidade do processo e na forma de cálculo da produção. Dessa forma, é importante reutilizar não somente os componentes, mas também os padrões de composição dos componentes que definem um tipo de instalação de produção de petróleo. A reutilização da composição dos componentes pode ser feita por meio da criação de *templates* representando um processo ou parte dele, o que equivale, na Petrobras, a um tipo de instalação de produção de petróleo. Como exemplo da utilização de *template* para representar um tipo de instalação pode-se ver na Figura uma representação de uma instalação do tipo Ponto de Coleta e a especificação de um *template* para modelar outras instalações desse mesmo tipo. O *template* define espaços para tipos definidos de equipamentos (representados na Figura por retângulos) e relações de fluxo entre eles.

Figura 22 – *Template* de um Ponto de Coleta



Fonte: Autoria própria do autor desta dissertação (2015).

Uma solução para criação de *templates* utilizando componentes foi proposta no contexto do Metalaboratório (GOMES, 2014), o *Active Web Template*, já

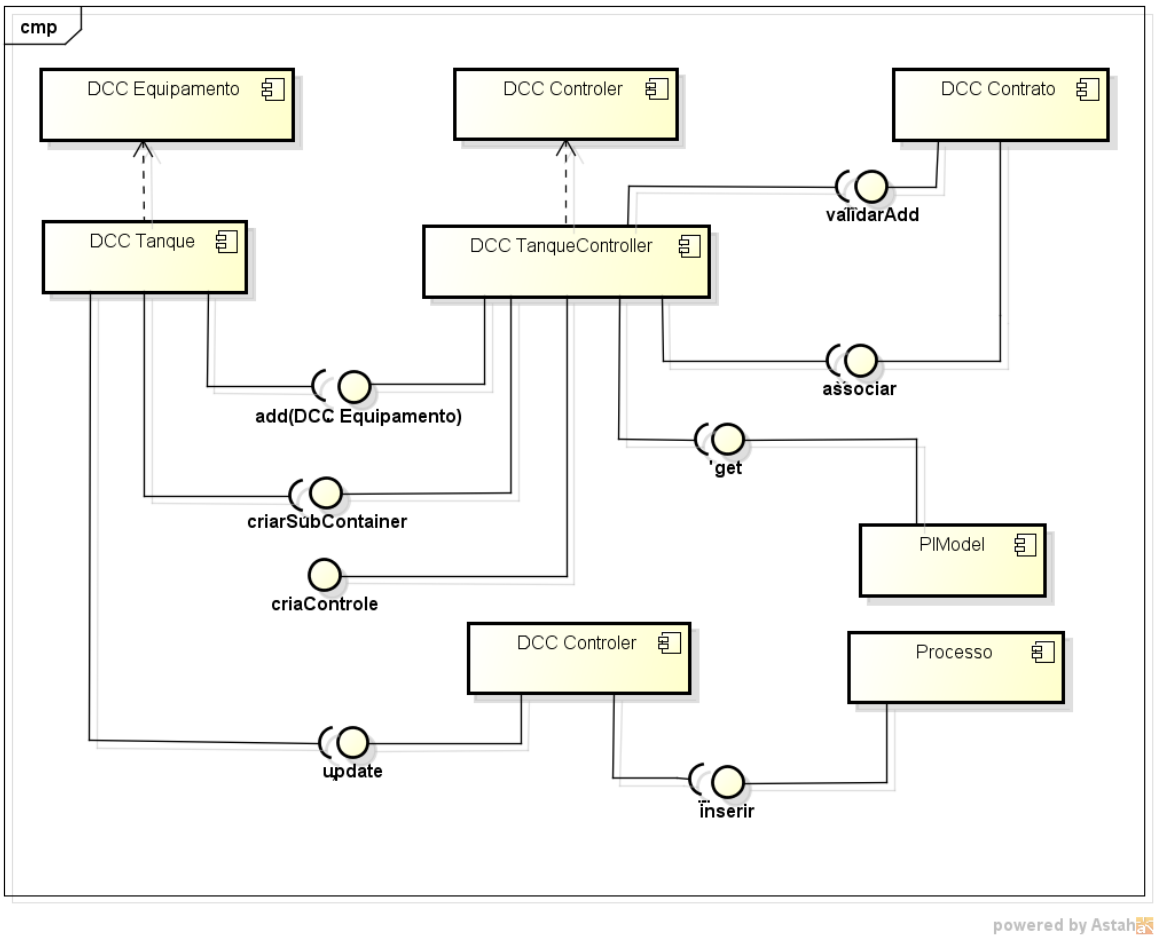
discutido no Capítulo 2. O *Active Web Template* pode ser visto como uma generalização de composições e estruturas de páginas Web, onde a estrutura do *template* é construída por um usuário através de uma formatação fixa e utilizando o meta-DCC para representar os pontos flexíveis de alteração do *template*.

Neste trabalho propomos uma família de componentes, denominada de Processo. Os componentes desta família possibilitam a estruturação de *templates* utilizando componentes de software. O componente Processo possibilita a criação de composições de componentes Equipamento pelos usuários, permitindo assim maior flexibilidade, já que todas as partes que compõem o *template* podem ser criadas e alteradas em tempo de execução em função das necessidades do usuário-autor.

O componente Processo foi inspirado na solução *Active Web Template* (GOMES, 2014). Ampliando o conceito adotado no *Active Web Template*, foi desenvolvido neste trabalho um conjunto de componentes que captura os padrões de composição e os *layouts* das instalações de produção de petróleo definidas pelo usuário autor.

A família de componentes para modelar o processo de uma instalação de produção é estruturada em quatro componentes: Processo, Container, Contrato e Layout (Figura 23 – Relacionamento da família de componentes de processo.). Cada um deles trata de aspectos diferentes do processo mapeado. A seguir apresentaremos cada um dos componentes propostos com um exemplo de utilização dentro processo de produção de petróleo terrestre. Na Figura 23 – Relacionamento da família de componentes de processo.podemos visualizar todos os componentes de processo e seus relacionamentos com os componentes de equipamento. Mais uma vez, adotamos a notação UML como uma abordagem visual para a representação das relações entre os componentes.

Figura 23 – Relacionamento da família de componentes de processo.



Fonte: Autoria própria do autor desta dissertação (2015).

4.1 COMPONENTE PROCESSO

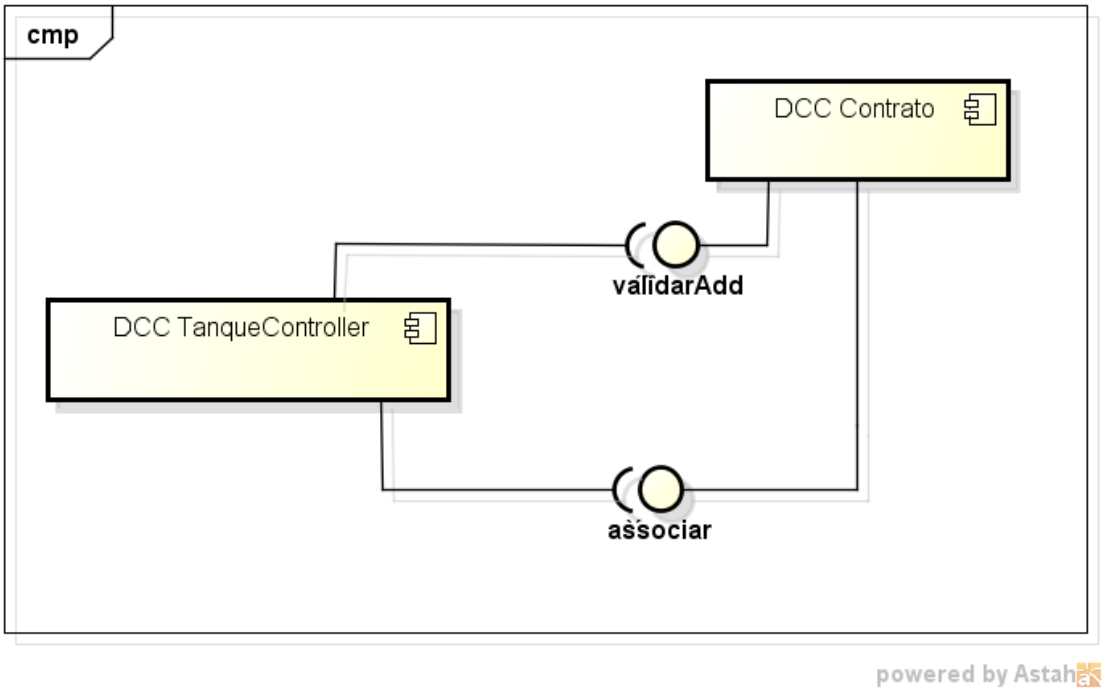
A disposição física e as relações entre os componentes que representam os processos de produção de petróleo e gás para efeito de monitoramento é construída também como componentes de software. Os componentes trabalham como um mecanismo de conexão para modelar as associações entre os componentes e como um mecanismo de layout indicando como a realização gráfica dos componentes será apresentada na tela do computador. Os componentes propostos não têm a intenção de apoiar todas as funcionalidades de uma modelagem de processo BPM, eles foram desenvolvidos para ajudar os usuários-autores durante a fase de criação de telas de monitoria para compor instalações de produção de petróleo consistentes. É função do *DCC Processo*, por exemplo, evitar ligações entre componentes que não podem ser ligados. Cada componente *Processo* pode representar um tipo de

instalação de produção de petróleo, no capítulo 5 são discutidos os tipos de instalações existentes na Petrobras e sua relação com o processo de produção.

O Componente *Processo* inicia a criação da planta e desenha no navegador o espaço onde será construído o monitor do processo. Para desenhar o componente Processo foi utilizado o elemento HTML *div*. O componente *DCC Processo* é inserido em uma página Web utilizado uma instrução simples, detalhada no capítulo 5, definida pelo framework *Componere* (HERMANO; SANTANCHÈ, 2015). Todo desenho do processo será construído a partir do componente Processo implementado.

Parte do código referente a implementação do componente *Processo* utilizando o framework *Componere* é mostrado na Figura 25 em três áreas distintas. Na área A são definidas as variáveis utilizadas e o componente do tipo Processo é instanciado utilizando a sintaxe do framework *Componere*. A variável *idProcesso* guarda a identificação da instalação de produção mapeada. Neste exemplo o identificador recebe a denominação de “Ponto de Coleta Exemplo”, indicando que trata-se de uma instalação simples, mas muito usual na Petrobras, ou seja, um Ponto de Coleta. Na área B da Figura a interface provida é declarada através do método *inserir* que recebe um componente do tipo *Container* como parâmetro. Finalmente na área C tem-se o denominado ciclo de instalação do framework *Componere*, onde o componente *Processo* é desenhado na tela do navegador e os elementos acessórios, que no caso especifico representam botões de ação, são criados para manipular os componentes que estão associados ao componente *Processo*. Uma atribuição componente Processo é desenhar na tela a delimitação do espaço onde os componentes serão inseridos e criar os elementos visuais que exercem alguma ação sobre os componentes internos ao processo, a exemplo de botões, links, caixas de seleção entre outros definidos pelo usuário em tempo de construção.

Figura 24 – Relacionamento do componente Contrato



Fonte: Autoria própria do autor desta dissertação (2015).

Figura 25 – Trecho de código da implementação de um componente *Processo*

A

```
var Processo;

Processo = new Component('./processo.js');

var idProcesso ='Ponto de Coleta Exemplo';

var nome='PontoColeta';
```

B

```
Processo.extend('./ui-element.js');

Processo.provide('processoInterface', function (done) {
  done({'inserir' : this.inserir});
});
```

C

```
Processo.install(function (done) {

    this.inserir = function (dcc) {
        $("#"+nome).append(dcc);
    };

  this.create = function () {

        var lHtml="<div id='"+nome+"' class=\"container\"</div>";

        //Escreve nome do processo na Tela do usuário
        identificarProcesso(idProcesso).

        //Criar Botão para alterar visão entre Processo BPMN x Monitor de Produção
        criarBotaoVisaoProcesso();

        //Criar Botão para salvar hierarquia de componentes em um arquivo XML
        criarBotaoSalvar();
```

Fonte: Autoria própria do autor desta dissertação (2015).

Além da definição do processo, o componente Processo deve representar também todos os componentes existentes na instalação. Os equipamentos do Processo são colocados em Container e componente Processo é construído como uma composição de componentes do tipo *Container* (Figura 20), que será apresentado na próxima seção.

4.2 COMPONENTE CONTAINER

O *Container* é um componente que representa um espaço totalmente fechado usado para armazenar outros componentes. Tipicamente, o *Container* armazena componentes do tipo *Equipamento* ou outros *Containers.*

Cada *Container* tem associado a ele componentes *Layout* e *Contrato*. O componente *Layout* armazena as regras utilizadas para organizar os componentes dentro do *Container* para fins de apresentação. O componente *Contrato* materializa regras, restrições e associação entre os componentes dentro de seu Container associado.

O *Container* é o principal componente dentre os propostos para representar um processo, pois ele determina a estrutura da composição de componentes. O componente Processo, discutido na seção anterior, define nominalmente a instalação mapeada, já os componentes *Container* utilizados definem visualmente a instalação de produção através das regras de associação que impõem sobre os DCC Equipamento que são inseridos nele. Todos os equipamentos inseridos são armazenados em uma estrutura de dados, o que permite persistência e posterior recuperação para reutilização.

Na Figura 26 um trecho de código de uma implementação do componente *Container* é mostrado. O código na área A define as variáveis de inicialização do componente, como já explicado para os demais componentes apresentados. Diferente dos outros componentes, o componente *Container* possui uma estrutura de dados que é utilizada para guardar referência para todos os componentes inseridos nele. As referências dos equipamentos são recuperadas da estrutura de dados através da identificação única do equipamento. No caso da Petrobras a TAG do equipamento é a identificação que serve a esse propósito. A estrutura de dados é armazenada e disponibilizada para os demais componentes através do conceito de

Local Storage (SCHOOLS, [s.d.]), uma propriedade disponibilizada na especificação do HTML5 que permite armazenamento local de dados a partir de aplicações Web.

Na área B do código da Figura 26 as interfaces providas e requeridas são declaradas. Os métodos *update* e *criaSubContainer* é disponibilizado para que outros componentes possam ser inseridos no *Container*. Duas interfaces são consumidas no componente *Container*, uma do componente Processo ligado ao *Container* e outra do componente Contrato relacionado a ele. Pode-se notar também no código da área B a implementação da publicação de um evento, chamado *timeUpdate.* Este evento é configurado para ser disparado a cada dez segundos e notifica todos os componentes que se inscreveram para escutá-lo.

Figura 26 – Trecho de código da implementação de um componente *Container*

A

```
var Container;

var elementos=[];

Container = new Component('./container.js');
```

B

```
Container.extend('./ui-element.js');

Container.publish('timeUpdate');

Container.provide('containerInterface', function (done) {
  done({'update' : this.update,'criaSubContainer':this.criaSubContainer});
});

Container.require('processoInterface', function (processo,done) {
  this.processo = processo;
  done();
});

Container.require('contratoInterface', function (contrato,done) {
  this.contrato = contrato;
  done();
});
```

Fonte: Autoria própria do autor desta dissertação (2015).

Na Figura 27 temos outro trecho de código com o ciclo de instalação do componente *Container*. No código da área A temos a implementação da interface *ContainerInterface* através do método *update,* que recebe um objeto e uma variável do tipo *String* como parâmetros. O objeto contém os atributos do equipamento que será inserido no *Container,* a exemplo do identificador (Tag) e da imagem que representará o equipamento para o usuário. A variável *String* é utilizada para

identificar o elemento HTML que será criado para desenhar o instrumento. Na área B temos a implementação de outro método disponibilizado através da interface *ContainerInterface,* o método *criaSubContainer* é utilizado para permitir a criação de componentes do tipo *Container* dentro do um componente *Container*.

Figura 27 – Implementação de componente *Container*

A

```
Container.install(function (done) {

    this.update = function (obj,str) {

        equipamento = JSON.parse(obj)
        elementos.push(equipamento.tag);
        var image2 = document.createElement('IMG');
        image2.src=equipamento.imagem;
        image2.id = equipamento.tag;

        $("<div class='ctDiv' id='" + equipamento.tag + "'></div>").appendTo("#"+str);
        $('#'+equipamento.tag).append(image2);

    };
```

B

```
    this.criaSubContainer = function (obj) {

        $("<div class='ctDiv' id='" + obj + "'></div>").appendTo(this.getNomeContainer

        contrato.associar(obj);

    };

    this.tick = function () {
      this.timeUpdate();
    };
```

Fonte: Autoria própria do autor desta dissertação (2015).

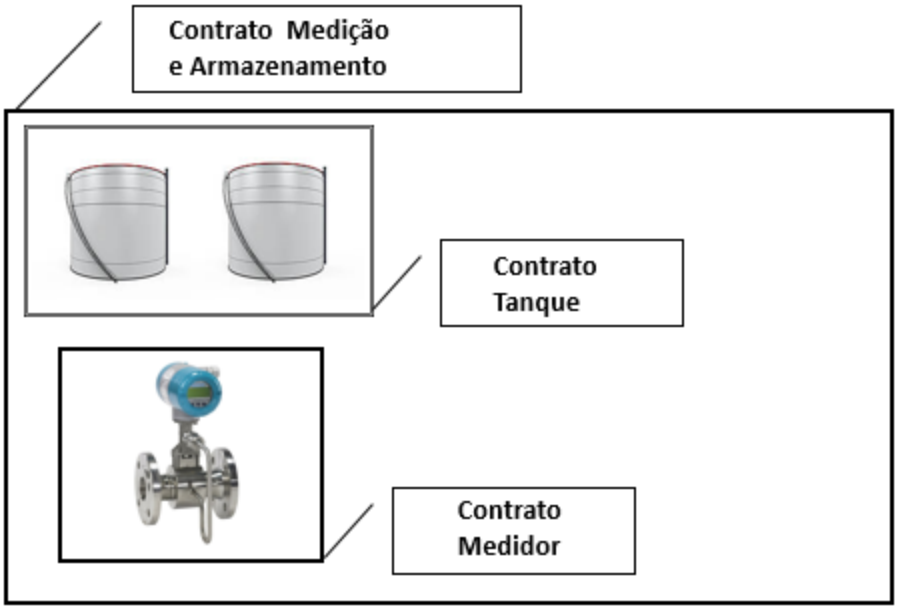
4.3 COMPONENTE CONTRATO

O *DCC Contrato* é o componente que materializa uma regra, restrição ou associação que o usuário-autor deseja registrar para a um *Container*. As regras definidas para o *Container* se aplicam também aos componentes *Container* que pertencem a ele.

O componente *Container* pode ser usado para agrupar componentes semanticamente relacionados. A semântica do *Container* é dada pelo componente *Contrato* associado a ele. É possível, por exemplo, ter *Containers* relacionados com etapas do processo, ou seja, *Container Produção*, *Container Medição e Armazenamento*, *Container Medição Auxiliar* e *Container Transporte*. O contrato

associado a estes recipientes restringe o tipo de equipamento que pode ser colocado no seu interior, um exemplo de uma regra de inserção. Considere, por exemplo, a representação da parte de uma instalação de produção de petróleo, onde algumas medições são feitas. Esta pequena parte da planta pode ser modelado como um *Container* relacionado com a classificação Medição e Armazenamento (Figura 28). Este tipo de *Container* só pode armazenar outros *Container* ou *Equipamentos* do tipo Medição e Armazenamento. Esta regra é estabelecida no componente *Contrato* associado a ele. No exemplo representado na Figura 28, uma *Container Medição e Armazenamento* armazena dois *Containers* do mesmo tipo, mas especializados com tipos de equipamentos diferentes dentre da classificação Medição e Armazenamento. O primeiro *Container* armazena dois componentes *Equipamento* do tipo *Tanque* e o segundo *Container* armazena um único componente *Equipamento* do tipo Medidor de Vazão.

Figura 28 – Exemplos de *Container*s tipificados



Fonte: Autoria própria do autor desta dissertação (2015).

O contrato de um Container representa mais do que as restrições de inserção dos seus membros. Um papel importante do componente *Contrato* é armazenar as possíveis relações entre os componentes internos, mapeadas em relações do tipo origem / destino. Considere, por exemplo, a construção de um Processo para monitorar o tipo mais simples de instalação: o Ponto de Coleta. Esta instalação pode ser modelada com um *Container* chamado *Ponto de Coleta*. Este *Container* tem três

outros *Containers* em seu interior: *Produção*, *Medição e Armazenamento* e *Transportes* (Figura 30). Esta configuração é imposta pelo *Contrato* do *Container Ponto de Coleta* que tem regras que limitam a inclusão para esses tipos de *Containers*. O *Contrato* do *Container Ponto de Coleta* tem regras de associação afirmando, por exemplo, que 1) o componente *Produção* pode ser relacionada com o componente de *Medição e Armazenamento*, 2) o componente de *Produção* deve ser a origem e o componente *Medição e Armazenamento* deve ser o destino.

Podemos ter mapeadas no componente Contrato regras do tipo associação, regras do tipo inserção e regras de fluxo. Para que o componente Contrato seja materializado é necessário que um modelo de instalação seja construído e persistido como um template de um tipo de instalação. Dessa forma todo o componente Contrato, que foi inteiramente construído a partir das composições realizadas entre os componentes Equipamento, é traduzido em um arquivo XML.

Figura 29 – Implementação de um componente Contrato

A

```
this.validarAdd = function (origem, destino) {

    if (origem.tipo === destino. tipo)

        return true;
    else

        return false;

};
```

B

```
this.associar = function () {

  $.ajax({
        type: "GET",
        url: "conexao.xml",
        dataType: "xml",
        success: function (xml) {
            $(xml).find('conexao').each(function () {
                var sOrigem = $(this).find('origem').text();
                var sNome = $(this).find('nome').text();
                var sDestino = $(this).find('destino').text();
                //Cria uma conexao visual entre dois componentes
                associarComponentes(sOrigem, sNome, sDestino);
            });
        },
        error: function () {
            alert("Ocorreu um erro inesperado durante o processamento.");
        }
        });
```

C

```
<conexao>
    <origem>Producao<origem>
    <nome>Produção<nome>
    <destino>Medicao<destino>
</conexao>
<conexao>
    <origem>Medicao<origem>
    <nome>Transferência<nome>
    <destino>Transporte<destino>
</conexao>
```

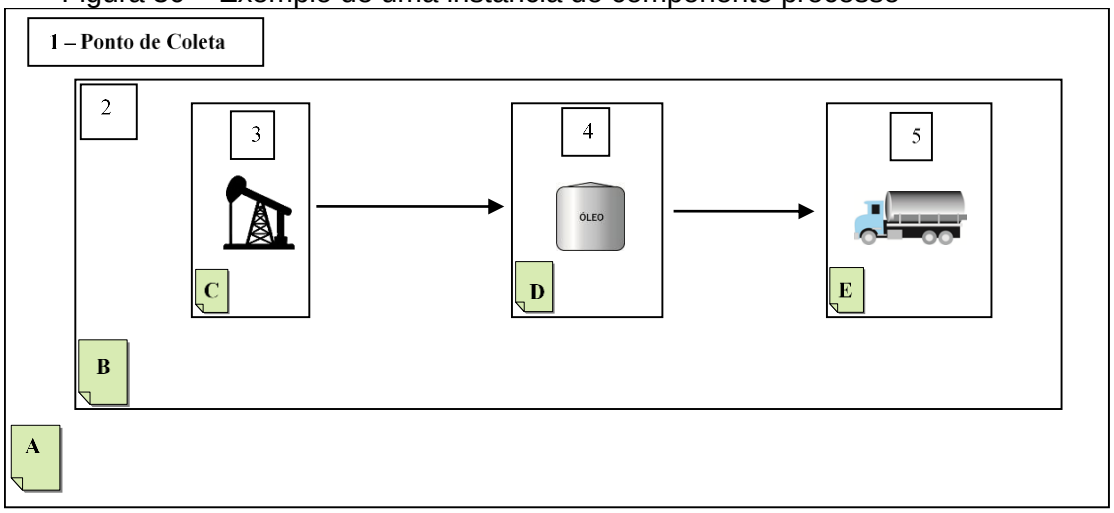
Arquivo Conexao.xml.

Fonte: Autoria própria do autor desta dissertação (2015).

Na Figura 29 um trecho de código de uma implementação do componente *Contrato* é mostrado divido em três partes: Em (A) temos uma implementação simplificada da validação de tipo realizada por um componente *Contrato*, executada ao inserir um componente *Equipamento* em um componente *Container*. O método recebe dois objetos JSON que possuem o atributo *tipo* preenchido com classificação

do componente. Em (B) é mostrado uma implementação do método que desenha na tela do usuário as associações definidas entre os componentes através de um arquivo XML mostrado em (C).

Figura 30 – Exemplo de uma instância do componente processo



Fonte: Autoria própria do autor desta dissertação (2015).

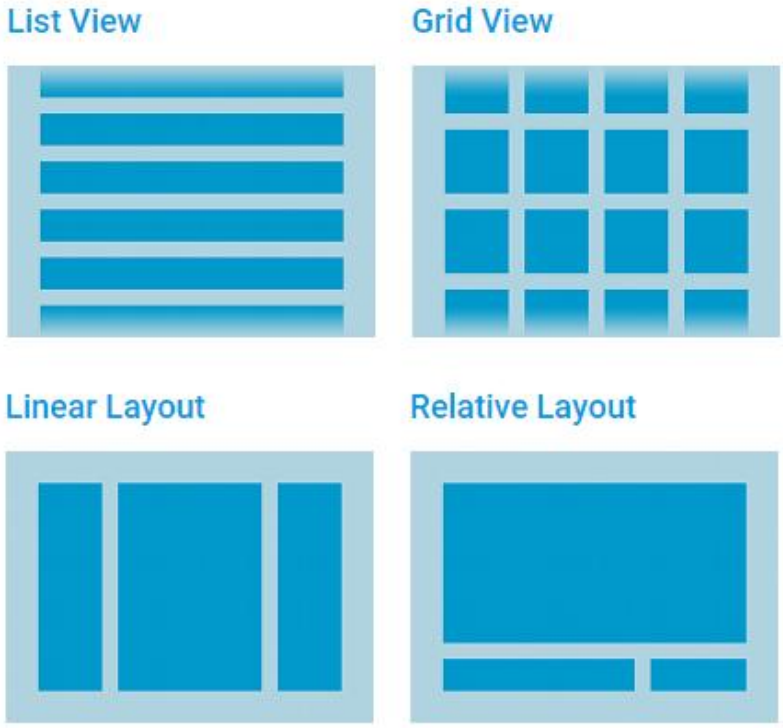
Na Figura 30 temos uma representação do processo Ponto de Coleta utilizando retângulos para ilustrar componentes *Containers* e *Processo*, ícones para ilustrar os componentes Equipamentos e setas para ilustrar os relacionamentos entre os componentes. O Componente *Processo Ponto de Coleta* (1) identifica o processo mapeado e o descreve brevemente. O *Contrato do processo ponto de coleta* (A) define que o componente *Processo* possui apenas um Container (2). Este *Container*, entretanto, possui três outros componentes do tipo *Container Produção* (C), *Medição* (D) e *Transporte* (E). Os componentes Container interno são guardados respeitando as posições informadas no componente Contrato, nesse caso foi definida uma regra que permite apenas associar três componentes *Containers* do tipo *Produção*, *Medição* e *Transporte* nessa ordem. O *Container Produção* (3) possui um *Equipamento Poço* associado a ele, o *Container Medição* (4) possui um *Equipamento Tanque* e o *Container Transporte* (5) com um Equipamento Carreta inserido nele. Os componentes *Contrato* desses compo*nentes* estabelecem que apenas componentes do tipo *Equipamento Produção*, *Medição* e *Transporte* podem ser inseridos nos respectivos Containers.

4.4 COMPONENTE LAYOUT

O Layout é componente responsável por definir a disposição gráfica dos componentes associado a um determinado componente *Container.* O *Layout* define o arranjo dos componentes na interface gráfica do usuário. Uma classe Container deve obrigatoriamente ter uma classe layout associada para definir a posição relativa entre os componentes que participam do Container.

A utilização do componente Layout foi inspirada nas definições de layout disponível na linguagem Java e no framework Android para dispositivos móveis (DEVELOPERS, 2015). Na Figura 27 temos alguns exemplos de possibilidade de implementação do componente Layout. No estudo de caso realizado para uma instalação do tipo ponto de coleta foram disponibilizados dois tipos de layout: o *linear layout* e *grid view*. A implementação do componente Layout será discutida no próximo capítulo.

Figura 31 – Exemplos de implementações de layout



Fonte: Developers (2015).

4.5 CONSIDERAÇÕES FINAIS

Este capítulo apresentou a família de componentes de Processo. Os componentes propostos permitem a construção de instalações de produção de petróleo e gás através da composição de componentes. O objetivo desta família de componentes é permitir que o usuário possa utilizar a composição de componentes para criar um repositório de instalações típicas do processo de produção de óleo e gás.

O componente *Contrato* é utilizado para representar as regras de associação entre componentes bem como estabelecer conjuntos semânticos de equipamentos através do componente *Container*, ou seja, tipificar um componente *Container* para que ele só seja associado a um tipo de equipamento de acordo com a classificação do equipamento. A família de componentes de Processo permite representação de composições de componentes como componentes de software. Desta forma, uma composição construída e salva por um usuário possa ser utilizada como *template* por outros usuários, que poderão realizar as alterações necessárias para adequar a composição utilizada para suas necessidades.

5 MONITORAMENTO DE PLANTAS INDUSTRIAS ATRAVÉS DA COMPOSIÇÃO DE COMPONENTES

Existem na Petrobras diferentes configurações do processo de produção de petróleo e gás terrestre, esses processos são mapeados em instalações de produção. A definição de processo adotada nesse trabalho diz respeito ao fluxo dos fluídos nos diversos equipamentos que compõem uma instalação e o resultado desse processo é a totalização do petróleo produzido.

A complexidade de uma instalação varia de acordo com fatores que influenciam na viabilidade técnica e econômica da produção dos poços da região e para iniciar sua operação a configuração da instalação precisa, necessariamente, ser aprovada pela agência reguladora de petróleo.

Podemos afirmar que a configuração de uma instalação é obtida através da distribuição dos equipamentos ao longo do fluxo dos fluidos resultantes da exploração de petróleo (RIBEIRO, 2003). Dessa forma, temos um tipo de instalação pré-definido estabelece um arranjo de equipamentos para atender uma finalidade especifica. De forma análoga, um template formado a partir de uma composição de componentes, que representam os equipamentos utilizados na produção de petróleo e gás, pode ser utilizado para descrever um tipo de instalação.

5.1 INSTALAÇÃO DE PRODUÇÃO DE PETRÓLEO

A Petrobras é uma empresa grande e complexa. Desta forma, é natural imaginar que a diversidade e complexidade de instalações é muito grande. No que se refere a produção terrestre de óleo e gás existem cinco tipos básicos de instalação: Ponto de Coleta, Estação coletora com tratamento, Estação coletora sem tratamento, Estação de tratamento de Óleo e Parques de Armazenamento.
Apresentamos a seguir apresenta-se uma breve descrição dessas instalações.

a) Ponto de coleta - Local onde está localizado um poço de petróleo e que tem pelos menos um tanque. Geralmente, o poço encontra-se em áreas isoladas e de difícil acesso. A configuração desse tipo de instalação é similar a uma estação coletora, mas a transferência de petróleo ocorre por transporte rodoviário.

b) Estação Coletora - Instalação com facilidades de produção de petróleo. Possui como função principal, separar a corrente de produção em gás e líquido, podendo ou não ser capacitada a separar o óleo da água. Uma estação coletora pode ser o destino da produção realizada em um ponto de coleta e esse tipo de instalação pode também ter capacidade de especificar o petróleo para as condições fiscais.
c) Estação de Tratamento de Óleo (ETO) – Uma ETO pode ser o destino do petróleo produzido por instalações do tipo Ponto de Coleta e Estações Coletoras. Este tipo de instalação é equipado com equipamentos de tratamento do petróleo suficientes para especificá-lo para as condições fiscais. Possui também a função secundária de armazenamento.
d) Parques de armazenamento - Instalação cuja função principal é o armazenamento de petróleo em uma rede de movimentação (transferência ou transporte) de petróleo. Como função secundária pode fazer um tratamento para melhoria da especificação do petróleo. Este tipo de instalação contém tanques de armazenamento de petróleo com a finalidade de receber, armazenar e transferir petróleo. O fato de estar recebendo petróleo diretamente de um ou mais campos produtores para posteriormente transferi-lo para o próximo ponto no processo qualifica a Estação ou Parque de Armazenamento de Petróleo como instalação de embarque ou desembarque de petróleo (IED).

O restante desse capítulo ilustra a criação de uma instalação do tipo Ponto de Coleta para efeito de monitoramento.

5.2 CRIAÇÃO DE MONITORES DE PRODUÇÃO ATRAVÉS DA COMPOSIÇÃO DE COMPONENTES

O objetivo deste trabalho é a criação de um ambiente que permita a construção de softwares, que monitoram e documentam uma instalação de produção de petróleo e gás, através da composição de componentes. Para que isso seja possível, foram criados alguns componentes de software representando tanto os equipamentos quanto os processos de produção de petróleo e gás. Nesse contexto, foram discutidos nos capítulos quatro e cinco a implementação dos componentes que servirão de base para a criação dos sistemas que irão monitorar e documentar uma instalação típica da Petrobras. A título de prova de conceito e como

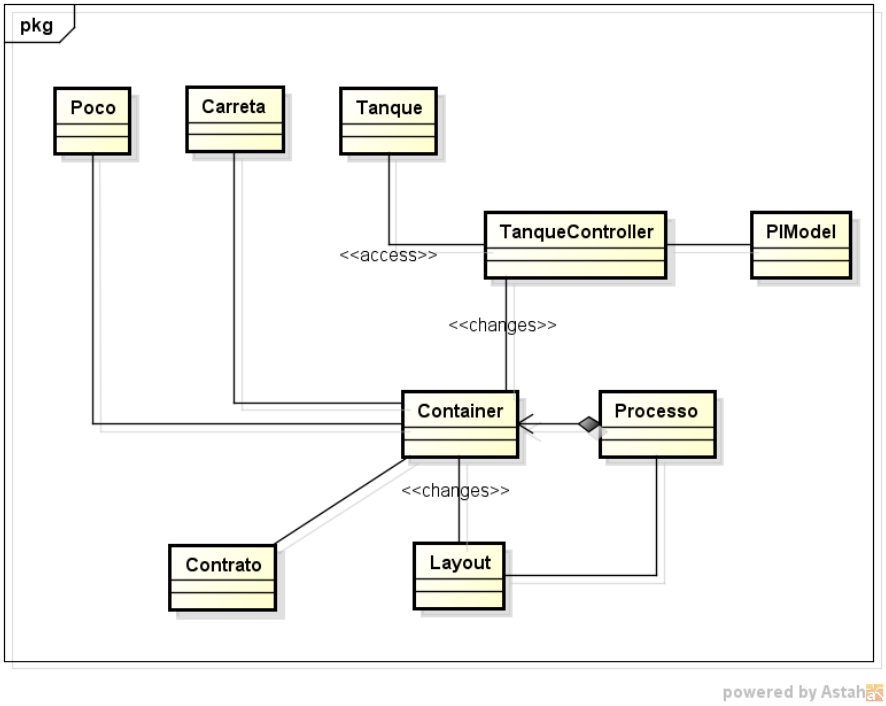
forma de validar os componentes desenvolvidos foi criado o monitor de uma instalação de produção do tipo Ponto de Coleta, que chamaremos de Monitor Ponto de Coleta.

Uma instalação do tipo Ponto de Coleta tem a configuração mais simples de produção, com poucos elementos e um fluxo elementar. O menor Ponto de Coleta possui um poço, um tanque para armazenar o fluido retirado do poço e, eventualmente, uma carreta para transportar o fluido produzido. Apesar de ter o fluxo simplificado, uma instalação do tipo Ponto de Coleta possui elementos suficientes para utilizarmos os componentes propostos neste trabalho.

Conforme discutido nos capítulos anteriores os componentes especificados foram especializados a partir do modelo de componente para Web, chamado DCC e implementados utilizando o framework *Componere* (HERMANO; SANTANCHÈ, 2015), que é um framework para construção de componentes Web utilizando uma implementação do DCC e a linguagem Javascript. A linguagem Javascript permite a execução dos componentes criados nesse framework em um navegador Web, sem a necessidade de nenhuma ferramenta adicional.

Na Figura 28 são exibidos os componentes utilizados na implementação do Monitor Ponto de Coleta. Nessa instalação temos os componentes *Processo*, *Container*, *Contrato*, *Layout*, *Tanque*, *TanqueController*, *Carreta* e *Poço*. Podemos observar na Figura 28 que o *DCC Container* é o componente central na implementação do Monitor Ponto de Coleta, pois ele é o responsável por desenhar os componentes na tela, seguindo as definições dos componentes que requisitam o desenho.

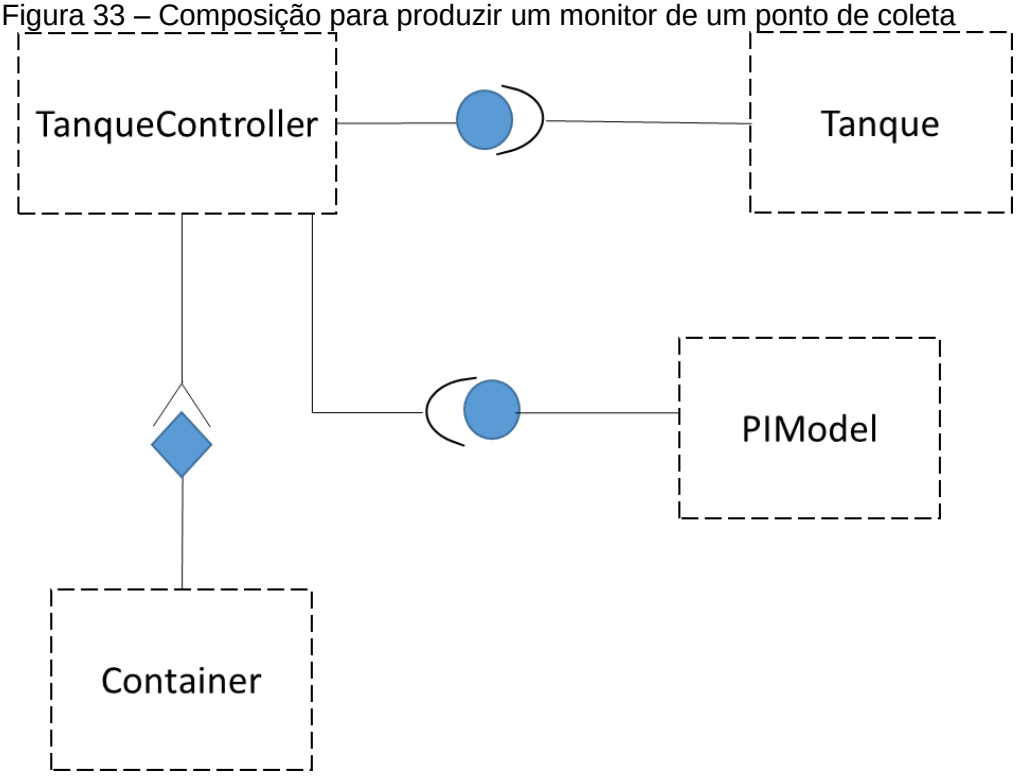
Figura 32 – Componentes utilizados para construir um ponto de coleta



Fonte: Autoria própria do autor desta dissertação (2015).

No ambiente de composição de componentes a comunicação entre componentes ocorre através de interfaces e eventos. Na Figura 29 temos um exemplo da utilização de interfaces e eventos na comunicação de componentes DCC. O componente *Tanque* requer uma interface provida pelo componente *TanqueController*, que utiliza essa interface para desenhar uma representação gráfica do tanque na tela do usuário. O *TanqueController* utiliza uma interface provida pelo componente *PIModel* para obter dados da automação de um determinado tanque. O componente *Container* publica um evento de atualização para todos os componentes inseridos nele, como por exemplo, o *TanqueController*, que está inscrito para receber mensagens desse evento e solicita novos dados do *PIModel* a cada notificação de atualização recebida. Os detalhes de funcionamento do Monitor Ponto de Coleta serão explicados a seguir.

Figura 33 – Composição para produzir um monitor de um ponto de coleta



Fonte: Autoria própria do autor desta dissertação (2015).

O DCC *Processo* inicia a criação da planta e desenha no navegador o espaço onde será construído o monitor. Para desenhar o componente Processo e o componente Container foi utilizado o elemento HTML Div. Na Figura 30 temos o código utilizado para inserir um componente Web em uma página HTML simples, segundo notação definida pelo framework *Componere Web-components* (HERMANO; SANTANCHÈ, 2015).

Figura 34 – Método de inserção de um componente na página HTML

```
<component type="./processo.js" id="processo1"></component>
```

Fonte: Autoria própria do autor desta dissertação (2015).

O DCC *Container* é uma instância de um componente do tipo *Container* e se configura como o principal componente para a construção do *Monitor Ponto de Coleta*. A especificação do componente *Container* permite a criação de componentes tipificados para receber apenas um determinado tipo de componente

de acordo com a classificação proposta neste trabalho. O componente *Container* que representa toda a instalação possui três componentes do tipo *Container* tipificados para receber componentes do tipo Produção, Medição e Armazenamento e Transporte. O *Container* utiliza uma interface provida pelo componente *Processo* para se desenhar na tela do navegador Web.

Para que uma interface provida por um componente possa ser utilizada por um componente que requisita essa interface, as instâncias dos componentes precisam ser conectadas, de acordo com a metodologia definida pele framework para criação e execução de componentes. Na Figura 31 pode-se ver um exemplo da ligação entre duas instâncias de componentes que se relacionam através de interfaces. Na inicialização do componente *Container* é informado como parâmetro a instância do componente que provê a interface por ele requerida.

Figura 35 – Instância de um componente com uma referência de interface

```
]<component type="./container.js" id="container1" processoInterface="processo1">
-</component>
```

Fonte: Autoria própria do autor desta dissertação (2015).

O DCC *Contrato* define os relacionamentos entre os componentes que compõem o Monitor Ponto de Coleta. No estudo de caso os relacionamentos foram representados por setas direcionais que indicam o sentido do fluxo do fluído. Os relacionamentos são construídos na implementação do componente Contrato. Outra atribuição do *Contrato* é a de tipificar o componente *Container* ao qual ele está associado, na implementação do Monitor Ponto de Coleta o componente *Container* aceita todos os tipos de equipamentos, com exceção do tipo Medição Auxiliar. Este tipo de equipamento não faz parte do modelo de instalação Ponto de Coleta.

O DCC *Tanque* cria a representação gráfica do equipamento Tanque na tela do navegador e inicia os atributos que serão monitoradas pelo aplicativo. O componente *Tanque* requisita a interface do componente do tipo Controle para criação de um painel onde as variáveis de interesse serão visualizadas, atualizadas e acompanhadas.

O DCC *TanqueController* é o responsável por monitorar as variáveis definidas na criação do componente *Tanque.* Ele provê uma interface para criação de *Container* onde o usuário poderá visualizar os atributos do tanque que serão

acompanhados. Neste componente é utilizado o padrão *Publish*/*Subscribe* em relação ao componente *Container*. O componente *Container* publica um evento, que é uma chamada de atualização para todos os componentes internos a ele, que se inscreveram para receber mensagens desse evento. Na Figura 36 temos o trecho de código onde o *Container* publica o evento "*timeUpdate*" que é disparado a cada dez segundos através do método *tick().*

Figura 36 – Trecho de código para publicação de um evento

```
Container.publish('timeUpdate');


    this.tick = function () {
      this.timeUpdate();
    };

  this.create = function () {
    done();
  });

  Container.start(function (done) {
  this.create();

  setInterval(this.tick.bind(this), 10000);

  done();
});
```

Fonte: Autoria própria do autor desta dissertação (2015).

Na Figura 37 temos o trecho de código em que o componente *TanqueController* se registra para receber mensagens dos eventos disparados pelo *Container* e a cada ciclo de chamada o TanqueController atualizar os atributos do Tanque monitorados na interface do usuário.

Figura 37 – Trecho de código mostrando

```
tqControle.listen('timeUpdate', function () {
  this.get();
});
```

Fonte: Autoria própria do autor desta dissertação (2015).

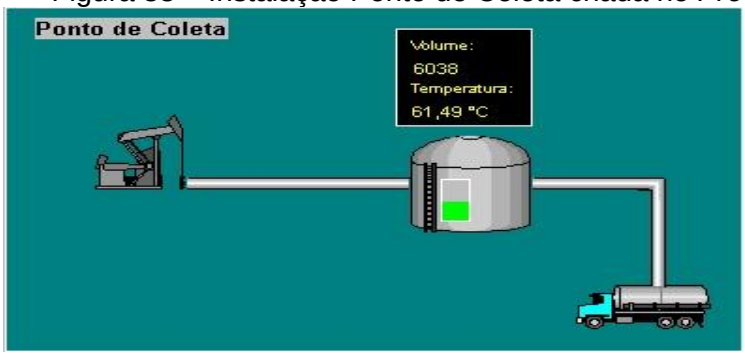
O DCC *PIModel* é o responsável por recuperar os dados monitorados do sistema PI. Os dados são recuperados através de um Web-service que acessa a base de dados do sistema historiador. Uma pequena aplicação que acessa os dados do PI via ODBC e disponibiliza via serviço web foi desenvolvida em Java para o presente estudo de caso.

DCC *Poço* ***e*** DCC *Carreta* tem a responsabilidade apenas de desenhar na tela do navegador a representação gráfica dos equipamentos Poço e Carreta, respectivamente.

DCC *Layout* define a organização dos componentes existentes em um determinado *Container*. Ao ser instanciado o *Layout* disponibiliza na interface do usuário uma caixa de seleção onde serão exibidos todos dos *Container*es existentes na tela de acordo com os parâmetros definidos na criação do componente Layout. No caso do Monitor Ponto de Coleta foram listados os *Container*es filhos de componente *Container.* Ao selecionar com *Container* o usuário pode alterar a disposição dos componentes pertencentes a ele de acordo com os layouts disponíveis: linha ou coluna.

A título de ilustração e comparação com os recursos existentes atualmente na Petrobras para criação de plantas de monitoramento, foi criada com no software ProcessBook uma instalação do tipo ponto de coleta (Figura 38).

Figura 38 – Instalação Ponto de Coleta criada no *ProcessBook*

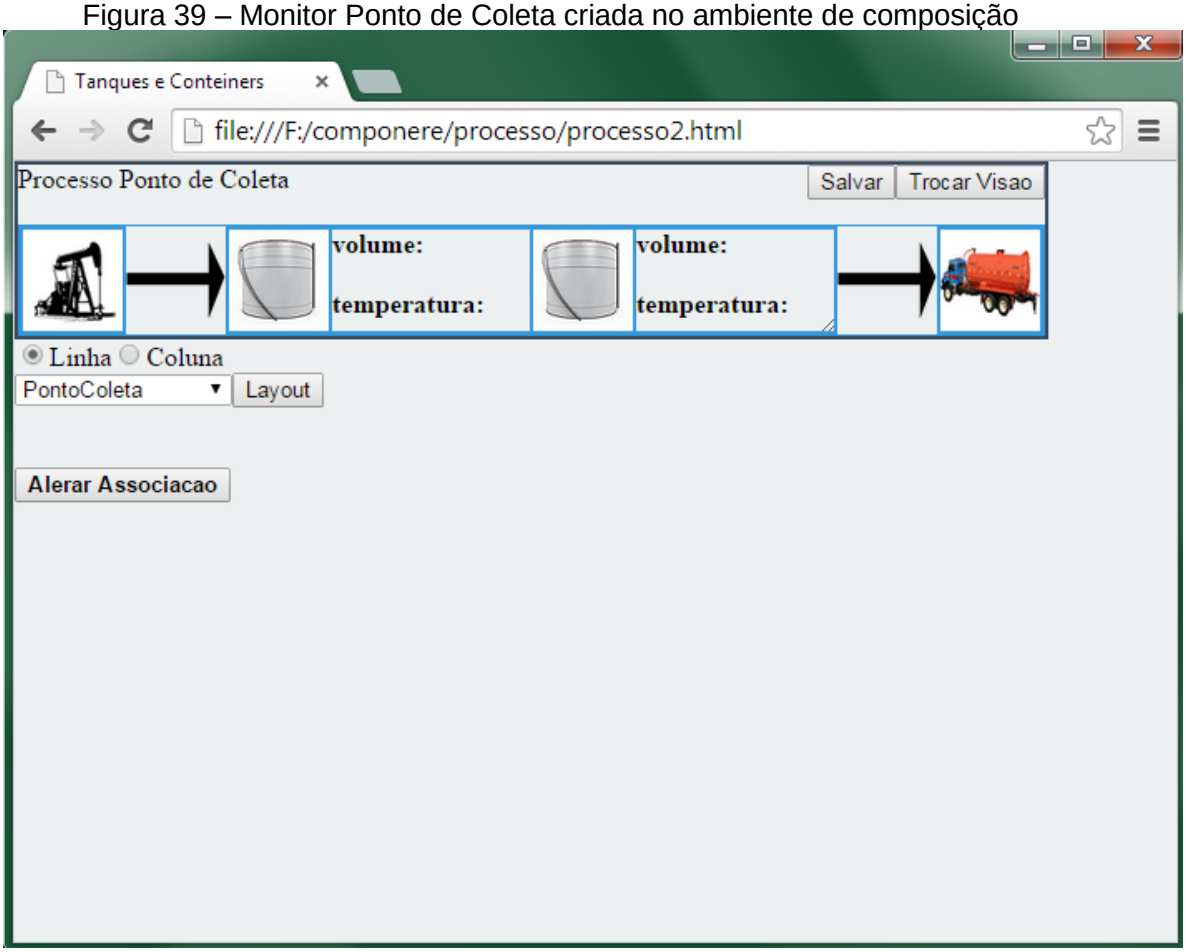


Fonte: Autoria própria do autor desta dissertação (2015).

Na Figura 39 temos o Monitor Ponto de Coleta criada como prova de conceito dos componentes propostos neste trabalho e para demostrar a montagem de uma instalação através da composição de componentes. A principal diferença nas duas abordagens é que no ProcessBook a criação de plantas é feita dentro de um determinado contexto, utilizando imagens pré-determinadas e restritas ao sistema PI

da empresa OsiSoft. A ferramenta proposta neste trabalho possui código aberto, os componentes podem ser implementados livremente e acoplados ao sistema sem necessidade de maiores configurações. As plantas são construídas através da composição de componentes, onde o usuário tem a liberdade de criação e utilização de qualquer repositório, tendo como ambiente de execução um navegador Web e como linguagem de programação o Javascript, ambos amplamente difundidos no meio profissional e acadêmico.

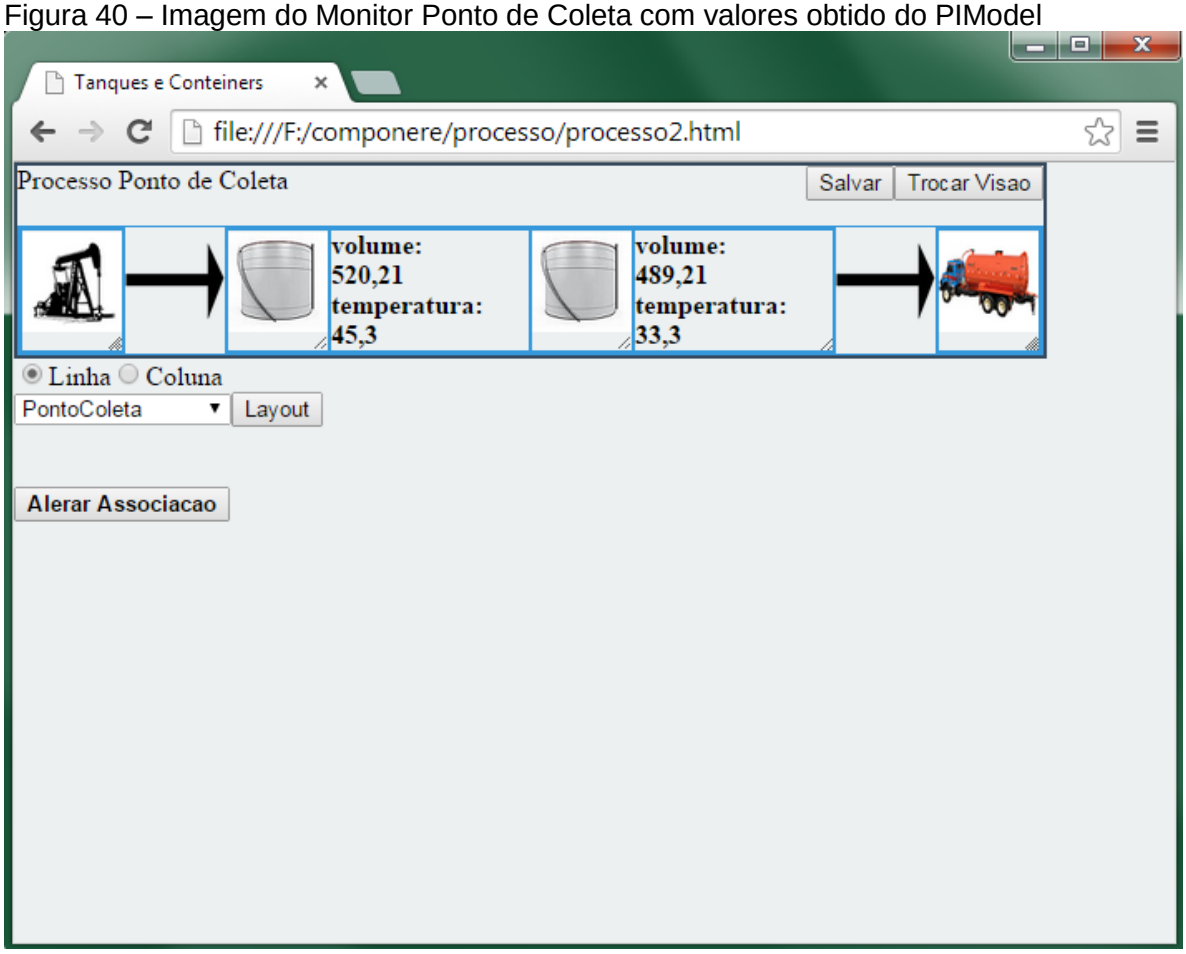
Figura 39 – Monitor Ponto de Coleta criada no ambiente de composição



Fonte: Autoria própria do autor desta dissertação (2015).

Na Figura 40 temos uma imagem do Monitor Ponto de Coleta com os dados carregados, obtidos através de interface com um banco de dados temporal da aplicação PIMS PI. Os dados são atualizados a cada dez segundos, de acordo com a definição configurada na criação do componente *Controle*.

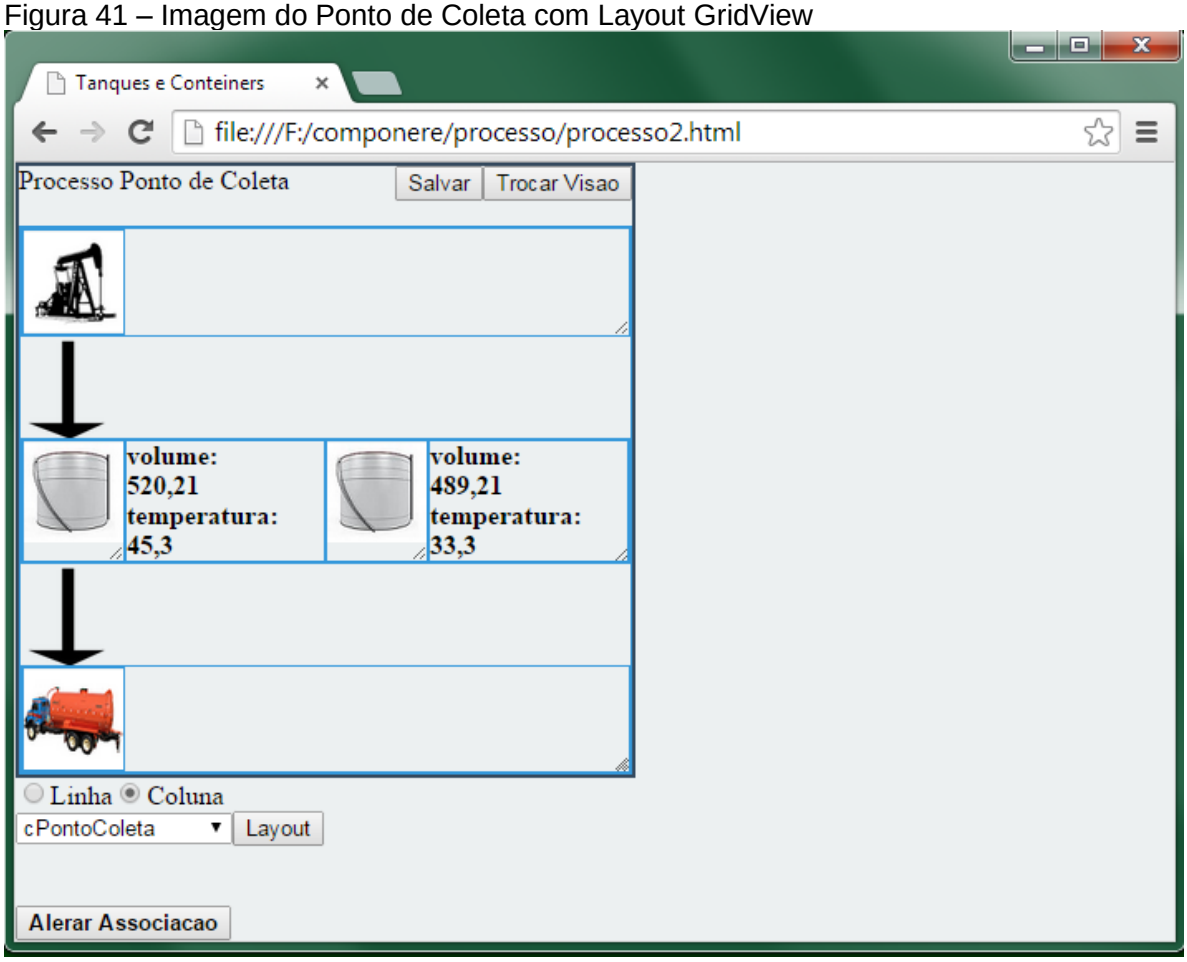
Figura 40 – Imagem do Monitor Ponto de Coleta com valores obtido do PIModel



Fonte: Autoria própria do autor desta dissertação (2015).

Na Figura 41 temos um exemplo da utilização do componente *Layout*. O comportamento dos componentes foi alterado para uma disposição em formato de colunas, de acordo com o layout GridView definido na construção do componente *Layout*.

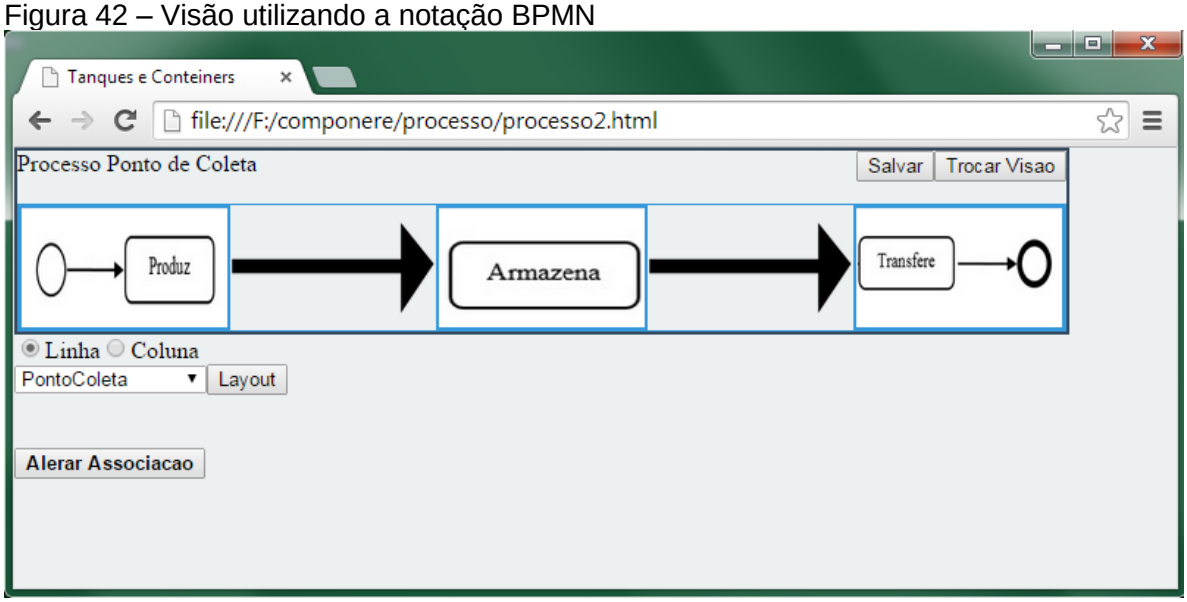
Figura 41 – Imagem do Ponto de Coleta com Layout GridView



Fonte: Autoria própria do autor desta dissertação (2015).

Na Figura 42 podemos ver a representação da instalação do tipo ponto de coleta utilizando a notação BPMN. O botão “Trocar Visão” criado pelo componente *Processo* invoca o método de cada componente *Container* associado a ele para que alterne sua representação gráfica para a notação BPMN. O componente *Container* que disponibilizar a visualização em BPMN deverá prover uma interface para essa ação que é requerida pelo componente *Processo* ao qual está associado.

Figura 42 – Visão utilizando a notação BPMN



Fonte: Autoria própria do autor desta dissertação (2015).

5.3 CONSIDERAÇÕES FINAIS

O objetivo deste capítulo é demonstrar a utilização dos componentes propostos para representar os diversos equipamentos existentes no processo de produção de petróleo e gás. Com a disponibilização dos componentes em um meio com livre acesso aos interessados, espera-se ter componentes cada vez mais complexos, adicionando novas funcionalidades para o ambiente de composição.

A visão de processo de negócio possibilita a utilização das composições de componentes realizadas como forma de documentação do processo de produção pela Petrobras. Foi escolhida uma notação padrão de representação de um processo, o BPMN, para permitir uma visualização clara e simplificada do processo de produção de petróleo e gás. Com essa visualização a Petrobras terá a opção de armazenar as equações que resultam no correto cálculo da produção de petróleo de forma gráfica, independentes de plataformas tecnológicas para utilização, podendo ser visualizada em um navegador da Web. A documentação do processo é atualizada pelo controle da produção no nível operacional através da visualização da composição de componentes como um monitor do processo, obtendo informações do equipamento em tempo real.

6 CONCLUSÃO E TRABALHOS FUTUROS

A principal motivação desse trabalho era a criação de uma infraestrutura de componentes de software para construção de sistemas que monitoram e controlam a produção terrestre de óleo e gás. As indústrias do setor utilizam sistemas construídos por empresas especialistas em automação que oferecem soluções proprietárias, monolíticas e que não permitem evolução por parte de terceiros. A proposta deste trabalho é disponibilizar uma aplicação extensível, com código aberto e disponível para utilização com fins comerciais e educacionais. Empresas fabricantes de equipamentos podem, no futuro, vender seus produtos já com a disponibilização de um componente que o representa em um repositório público para as empresas do setor que estejam interessadas em consumi-los.

Neste trabalho foi desenvolvido um ambiente de composição de componentes para modelar interfaces que monitoram e controlam uma instalação de produção de petróleo e gás. Este ambiente é baseado no ambiente para criação e execução de componentes Componere e em alguns conceitos utilizados no sistema Metalaboratório. O ambiente se configura como uma proposta alternativa de código aberto para as ferramentas PIMS que atualmente permitem a criação de telas para monitoramento de plantas, acessando dados da automação.

A contribuição principal deste trabalho pode ser dividida em três partes. Primeiro, foi proposta uma classificação dos equipamentos utilizados na produção de petróleo e gás de acordo com a etapa do processo em que os componentes se inserem. Esta classificação é importante porque é utilizada pelos componentes de software para representar regras e restrições do domínio de aplicação. Segundo, adotamos o modelo DCC para representar uma família componentes de equipamentos e do processo de produção de petróleo e gás. A família de componentes de Equipamentos fornece a representação física do equipamento para fins de apresentação, permite que o componente se conecte com qualquer fonte de dados com o propósito de gerar informação para o usuário e produzir e exibir valores para fins de monitoramento. A família de componentes de Processo é responsável por orquestrar a apresentação gráfica do equipamento e por moldar grupos semânticos relacionados. O componente Processo permite também modelar restrições e associações entre componentes. Em terceiro lugar, foi desenvolvido um

ambiente para construir composições que monitoram instalações de produção de petróleo e gás com base em componentes de software. Sua principal vantagem é a de ser executado no navegador, sem a necessidade de qualquer outra aplicação para o seu funcionamento.

Como trabalho futuro, pretende-se construir uma biblioteca de componentes do tipo equipamento, com os equipamentos comumente utilizados nas instalações de produção de petróleo e gás, disponibilizando os componentes implementados em um repositório. O repositório terá livre acesso na Web para os usuários interessados utilizarem na construção de suas plantas através do ambiente de composição. Para o ambiente de composição de componentes, pretende-se evoluir para um ambiente de autoria completo com uma interface para manipulação direta.

O desenvolvimento da aplicação completa permitirá que a funcionalidade do *DCC Contrato* de tipificar um componente *Container*, efetuando a validação do tipo de componente que pode ser inserido nele durante a construção de um monitor de processo, possa ser utilizada pelos usuários em tempo de criação. Pretende-se também que a ferramenta permita a construção de novos componentes Web estendendo, ou não, os componentes disponíveis na biblioteca.

REFERÊNCIAS

ALVES, F. J. Um modelo de repositório referência de componente de software (RRCS). **Revista Científica da Faculdade Atenas**, n. 2011, 2011.

BITTENCOURT, I. B. et al. Sistemas de Autoria para Construção de Ambientes Interativos de Aprendizagem Baseada em Agentes. **Revista Brasileira de Informática na Educação**, v. 15, n. 1, 2007.

BOARETTO, N. **Sistemas Supervisórios**. Joinville: Instituto Federal Santa Catarina, 2008.

BRAUNS, B. Viabilidade técnica e econômica na exploração de petróleo em campos maduros. Uma porta para a indústria nacional. In: **C**ONGRESSO NACIONAL DE EXCELÊNCIA EM GESTÃO ENERGIA, INOVAÇÃO, TECNOLOGIA E COMPLEXIDADE PARA A GESTÃO SUSTENTÁVEL, 2010. **Anais...** 2010. p. 1–19.

BRUNETON, E. et al. An Open Component Model and Its Support in Java. In: COMPONENT-BASED SOFTWARE ENGINEERING INTERNATIONAL SYMPOSIUM, CBSE (2004), 7., 2004. **Proceedings…** 2004. v. 674, n. x, p. 7–22,

CARVALHO, F. DE; FONSECA, M. D. O.; FILHO, C. S. **Sistemas Pims-Conceituaçâo, usos e beneficios**, 2005. Disponível em: <http://www.abmbrasil.com.br/materias/download/2490.pdf> Acesso em: 10 Jn. 2015.

COULSON, G. et al. A generic component model for building systems software. **ACM Transactions on Computer Systems**, v. 26, n. 1, p. 1–42, 1 fev. 2008.

COUNCILL, B.; HEINEMAN, G. T. Definition of a Software Component and Its Elements. **Open Distributed Systems**, p. 5–19, 2007.

COX, P. T.; BAOMING SONG. A formal model for component-based software In: IEEE SYMPOSIA ON HUMAN-CENTRIC COMPUTING LANGUAGES AND ENVIRONMENTS. 20011. **Proceedings...** 2001. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=995278> Acesso em: 10 Jn. 2015.

CRNKOVIC, I.; STAFFORD, J.; SZYPERSKI, C. Software **Components beyond programming.** [S.l.]: [s.n.], 1968.

D'SOUZA, D.; WILLS, A. **Objects, components, and frameworks with UML:** the catalysis approach. [S.l.]: Addison Wesley, 1998.

DE SORDI, J. O. **Gestão por processos:** uma abordagem da moderna administração. São Paulo: Saraiva, 2008.

DEVELOPERS, A. **API Android**. Disponível em: <http://developer.android.com/guide/topics/ui/declaring-layout.html>. Acesso em: 20 ago. 2015.

DOWLING, J.; CAHILL, V. Dynamic Software Evolution and The K-Component Model. In: WORKSHOP ON SOFTWARE EVOLUTION (OOPSLA 2001), 2001. **Proceedings...** 2001.

ECMA INTERNATIONAL. ECMA-404: The JSON Data Interchange Format (1st Edition). n. oct., 2013.

FONSECA, F. R. DA. Tecnologias de comunicação utilizadas em sistemas SCADA para supervisão de redes de abastecimento. **Revista Intech,** n. 87, p. 111–112, 2007.

FUNDAMENTAIS, A. Petróleo: Visão geral e aspectos fundamentais nas relações internacionais, v. 23, p. 129–147, 2013.

GOMES, A. **Web Metalaboratory**. São Paulo: UNICAMP, 2014.

GOMES, A.; HERMANO, R. E.; SANTANCHÈ, A. Web Metalaboratory: Composition of Laboratories on the Web. In: WORKSHOP ON TOOLS AND TECHNOLOGIES FOR VIRTUAL LABORATORIES 2014. **Proceedings...** 2004. p. 43–48.

GOMES, H. P. Automação e controle. In: SISTEMAS de Bombeamento–Eficiência Energética. 1. ed. João Pessoa: Editora Universitária - UFPB, 2009. p. 203–250.

HERMANO, R. E.; SANTANCHÈ, A. **web-components/componere**. Disponível em: <https://github.com/web-components/componere>. Acesso em: 18 jul. 2015.

HOPKINS, J. Component primer. **Communications of the ACM**, v. 43, n. 10, p. 27–30, 2000.

NETTO, F. S. **Medição de desempenho do gerenciamento de processos de negócio - BPM no PNAFE**: uma proposta de modelo. [S.l.]: [s.n.], 2006.

NUNES, A. V. **Avaliação de softwares de autoria**. Santa Catarina: Universidade Federal de Santa Catarina, 1997.

OSI SOFTWARE INC. **PI Process Book User Guide**. Disponível em: <http://elearn.viewcentral.com/content/ositraining/669eac132653ondemandmenu/ondemandmenu>. Acesso em: 18 jul. 2015.

PAIOLA, C. E. G.; TEIXEIRA, S. Aplicação de sensor virtual em processo de saneamento. **Revista Intech número 135**, p. 21–30, 2011.

RIBEIRO, M. A. **Medição de petróleo e gás natural**. 2. ed. Salvador: Petrobras, 2003.

SANTANCHÈ, A. **Fluid Web e componentes de conteúdo digital:** da visao centrada em documentos para a visao centrada em conteúdo. São Paulo: PhD

thesis, IC-UNICAMP, 2006.

SANTANCHÈ, A. et al. Componere - Autoria na Web baseada em Componentes. In: SIMP. BRASILEIRO DE SISTEMAS MULTIMÍDIA E WEB, 2009. **Anais...** 2009. p. 45–54.

SANTANCHE, A.; MEDEIROS, C. B. Geographic digital content components. **GeoInfo**, 2004.

SCHEUER, A. **Instalação e administração do sistema PI na Unidade Multipropósito de FCC**. [S.l.]: [s.n.], 2004.

SCHOOLS, W. **HTML5 Local Storage**. Disponível em: <http://www.w3schools.com/html/html5_webstorage.asp>. Acesso em: 1 jul. 2015.

SEIXAS FILHO, C. **PIMS -Process Information Management System:** uma introdução. Belo Horizonte: UFMG, 2011.

SENAI-PE. **Controladores lógicos programáveis**. Recife: SENAI-PE, 2008.

SILVEIRA, P. R. DA; SANTOS, W. E. **Automação e controle discreto**. São Paulo: Erica, 1998.

SMITH, H.; FINGAR, P. **Business process management:** the third wave. Tampa: Meghan-Kiffer Press, 2002. v. 1

SZYPERSKI, C.; DOMINIK, G.; STEPHAN, M. **Component software:** beyond object-oriented programming. 2. ed.. New York: ACM Press, 1998.

TSUTIYA, M. T. Automação de sistemas de abastecimento de água. In: ABASTECIMENTO de água. São Paulo: Departamento de Engenharia Hidráulica e Sanitária da Escola Politécnica da Universidade de São Paulo, 2004. p. 577–643.

VIANNA, W. DA S. **Controlador lógico programável.** [S.l.]: [s.n.], 2008.

WHITE, S. A. Introduction to BPMN. **BPTrends**, p. 1–11, 2004.

ZAMPRONHA, R. Sistemas Supervisórios. **Revista Intech,** n. 83, p. 9–16, 2006.